PhD Thesis

Semi-Supervised Ensemble Methods for Computer Vision

Christian Leistner

Graz University of Technology
Institute for Computer Graphics and Vision

Thesis supervisors
Prof. Dr. Horst Bischof
Dr. Vincent Lepetit

Graz, June 2010

When we write programs that "learn", it turns out that we do and they don't!

---

freely adapted from Yales's ML quotes

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Acknowledgement

To the god of thunder and rock'n'roll . . .

This thesis is the result of deep collaboration, discussion and support of many great people. My deepest gratitude goes to my supervisor Horst Bischof for giving me the chance to perform research on the exciting field of computer vision. Horst guided and supported my research but gave me the freedom to develop and follow my own ideas, even when he was not going conform with all my chosen research paths. It is not least the result of Horst's remarkable scientific management style and leadership, which makes the ICG the very pleasant and friendly but simultaneously professional working place it is today. I would also like to thank my second adviser, Vincent Lepetit for his comments on the thesis.

Thanks to my dear friend Amir, the dude. Amir totally changed my way on how to approach scientific problems and challenges. He has the remarkable skill of explaining even the most complicated algorithms and methods in a way that everybody immediately sees the quintessence and he is always patiently listening to all of my sometimes outlandish and bloodcurdling ideas and questions. Amir is a great friend and sportsman with a spirit driven by metal and rock'n'roll. I would also like to thank Helmut Grabner for showing me that research is all about discussion and that nothing else can lead to more profound ideas than beer. Great thank goes also to Peter M. Roth, the boss, who is not only a great collaborator but also a friend who supported me throughout my entire thesis. I would also like to thank Martin Godec, Papa G. in da house, who is the creator of BOB and, as the leader and co-founder of the ICG's noodle group, has also virtuous skills in the kitchen. Thanks to my buddy, soccer-master, guitar virtuoso and GPU-guru Jakob Santner. I would further like to thank: Werner Trobin, Bernhard Zeisl and everybody else of ICG's vision group, and also the often forgotten non-scientific staff, Renate, Manuela, Christina and Andi.

Most of all, I would like to thank my family: my mother, father, sister, my grand parents, my step-father, and my parents in law. Last but not least, I would like to thank my fiancée, Eva. Thanks for your love and support and for being on my side for over a decade.

This thesis was created at the Institute for Computer Graphics and Vision at Graz University of Technology between 2007 and 2010. During this time, the work was supported by the Austrian FFG project EVis (813399) under the FIT-IT prgrom and the Austrian Science fund (FWF) under the doctoral prgram Confluence of Vision and Graphics W1209. Since December 2008 the author has been employed as a research and teaching assistant at the Institute of Computer Graphics and Vision.

Abstract

Visual object classification and tracking are two of the cardinal problems in computer vision. Both tasks are extremely complicated and far from being solved. Recent advances towards building better detection and tracking systems were mainly achieved by improved representations and applying better learning algorithms. For the learning, usually supervised algorithms are applied which demand large amounts of hand-labeled data in order to yield accurate results. However, hand-labeling is a tedious and time-consuming task, which is also prone to human errors. Additionally, learning only from labeled data is not natural and there exist tasks, such as tracking-by-detection, where the learners have to be able to exploit both labeled and unlabeled data. Also, the growing number of digital images present in the web and off-line databases makes human hand-labeling hardly possible. Hence, we need learning techniques that are able to exploit huge amounts of unlabeled data with a reduced quantity of human interaction. These considerations have led to an increased interest in semi-supervised learning methods that learn from a small amount of labeled data and a large amount of unlabeled data.

In this thesis, we propose several novel approaches to semi-supervised learning using ensemble methods, such as boosting or random forests, and show their applicability on various computer vision tasks. The reasons for studying ensemble methods is that they are powerful and fast and are already used in many computer vision applications. In the first part of the thesis, we propose a novel semi-supervised boosting algorithm based on visual similarity learning and demonstrate its applicability to object detection. Then, we extend the method to on-line learning. In the second part of the thesis, we propose a novel random forest method that is able to learn from both labeled and unlabeled data. We demonstrate the benefits of the approach on both several machine learning tasks and object categorization where our method is able to train one inherent multi-class classifier rather than several one-versus-all classifiers. In the subsequent chapter, we demonstrate how to extend random forests to on-line mode. We also show how to apply random forests to a learning paradigm that is very similar to semi-supervised learning, *i.e.*, multiple instance

learning. Finally, we hypothesize that visual object tracking can be formulated as a one-shot-semi-supervised learning task. In the tracking experiments, we demonstrate that applying the proposed on-line semi-supervised and multiple instance learning methods lead to more stable and higher accurate tracking results.

Kurzfassung

Visuelle Objekterkennung und -verfolgung gehören zu den wichtigsten Aufgaben der Computer Vision. Für beide Problemstellungen gibt es eine grosse Anzahl praktischer Anwendungen, beide sind jedoch weit davon entfernt als gelöst betrachtet zu werden. Die in den letzten Jahren auf diesen Gebieten erreichten Fortschritte basieren meistens auf der Entwicklung von einerseits besseren Möglichkeiten Objekte zu beschreiben, zum Beispiel in Form von Mustern oder Merkmalen, und andererseits auf der Entwicklung und der Anwendung besserer maschineller Lernverfahren. Als Lernalgorithmen werden meistens vollständig überwachte Methoden verwendet, die üblicherweise eine grosse Anzahl an hand-gelabelten Daten benötigen um ausreichend akkurate Resultate zu liefern. Vollständig überwachte Verfahren haben jedoch das Problem, dass es meist schwierig und sehr aufwändig ist an gelabelte Daten zu gelangen. Ausserdem leben uns biologische Systeme vor, dass in der Natur Lernen zu einem Grossteil auf der Grundlage von ungelabelten Daten basiert. Zusätzlich führt die wachsende Anzahl von Bildern im Internet und in Datenbanken dazu, dass hand-basiertes Labeln von Daten immer schwieriger wird. Moderne Systeme, die in der Praxis zur Anwendung kommen, benötigen also Algorithmen, die in der Lage sind, von grossen Mengen an ungelabelten Daten zu lernen. Diese Überlegungen haben dazu geführt, dass die Forschung im Bereich des maschinellen Lernens sich in letzter Zeit vermehrt auf die Entwicklung von sogenannten halb-überwachten Lernverfahren konzentriert. Solche Verfahren sind in der Lage sowohl von einer kleinen Menge von gelabelten Daten als auch von einer grossen Menge an ungelabelten Daten zu lernen.

In dieser Doktorarbeit präsentieren wir mehrere neue Methoden des halb-überwachten Lernens basierend auf sogenannten Ensemble Verfahren, wie zum Beispiel Boosting oder Random Forests, und zeigen deren Anwendbarkeit auf einer Vielzahl von Computer Vision Applikationen. Die Motivation sich auf Ensemble Verfahren zu konzentrieren basiert auf der Tatsache, dass diese Algorithmen sich als sehr effizient erwiesen haben und bereits in sehr vielen Computer Vision Problemen zur Anwendung kommen. Im er-

sten Teil dieser Arbeit schlagen wir einen neuen halb-überwachten Boosting Algorithmus vor, der zusätzlich paar-weise Distanzfunktionen zwischen Bildern lernt und evaluieren diesen Ansatz auf Objekterkennungsproblemen. Zusätzlich zeigen wir, wie man diese Methode erweitern kann, damit sie on-line lernfähig ist. Im zweiten Teil dieser Arbeit beschäftigen wir uns mit Random Forests und präsentieren einen Algorithmus, der sowohl von gelabelten als auch von ungelabelten Daten lernen kann. Wir demonstrieren die Vorteile dieser Methode, wie inherente mehrklassenfähigkeit und erhöhter Lern- und Evaluierungsgeschwindigkeit, sowohl an Beispielen des maschinellen Lernens als auch an visueller Objektkategorisierung. Darauffolgend zeigen wir, wie man mit Random Forests an sequentiell eintreffenden Daten, also on-line, lernen kann. Weiters zeigen wir, wie man Random Forests für ein nah-verwandtes Gebiet des halb-überwachten Lernens, nämlich Multiple Instance Learning, anwenden kann und präsentieren einen neuen Algorithmus, MILForests. Im experimentellen Teil zeigen wir, dass MILForests in der Lage sind state-of-the-art Multiple Instance Ergebnisse zu erziehlen, dabei jedoch viel schneller sind als konkurrierende Verfahren.

Schlussendlich befassen wir uns mit dem konkreten Problem der visuellen Objektverfolgung und zeigen, dass Objektverfolgung als halb-überwachtes Lernproblem formuliert werden kann. Wir können daher die in dieser Arbeit vorgeschlagenen on-line halb-überwachten Algorithmen für Objektverfolgung verwenden. In den Experimenten zeigen wir, dass der Einsatz unserer halb-überwachten Methoden zu stabileren Objektverfolgungsresultaten führt.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

"What does the frog's eye tell the frog's brain?" - Lettvin *et al.* [Lettvin et al., 1959] asked themselves this question in the 50's of the 20th century. At the first glance, for many of us this question might not be relevant and we might rather wonder what drove Lettvin *et al.* to spend years if not decades to investigate a frog's visual physiology. However, by taking a closer look, we can observe how remarkable the skills of this tiny animal are: A frog searches prey by using its eyes, seeks it and in fractions of a second catches it with its tongue. Furthermore, it is able to survive from its predators by visually recognizing enemies and escaping from them. Hence, one can say that vision is probably the most important sense for a frog and without its eyes survival would probably not be possible. But how does this work? How is an animal as simple as a frog able to perform this remarkable visual tasks? Of course, the answer to that question is by no means trivial, but from a biological perspective, one can imagine that it is simpler to study these mechanisms on a frog that has a rather simple cognitive visual system compared to higher order species such as mammals; and this was also the reason why Lettvin *et al.* chose to investigate basic perceptual physiology using frogs, *i.e.*, due to their simplicity. Stepping forward from the 50's of the last century to the present, *i.e.*, 2010, science has made significant progress in understanding visual processes and not at least due to the groundbreaking work of [Lettvin et al., 1959] and other perceptual physiologists [Palmer, 1999, Livingstone, 2008] we do not only have reasonable insights on what the frog's eyes tell the frog's brain but also started understanding what human eyes tell the human brain. Even more, besides investigating the functionality of cognitive vision, nowadays the rapid development of computational technologies and digital cameras has allowed researchers and engineers to take over a more active role and investigate in building machines that are able to analyze and understand the world using visual intelligence such as humans. The research discipline that is concerned with developing mathematical techniques and algorithms in order to understand digital images and videos is called computer vision [Marr,

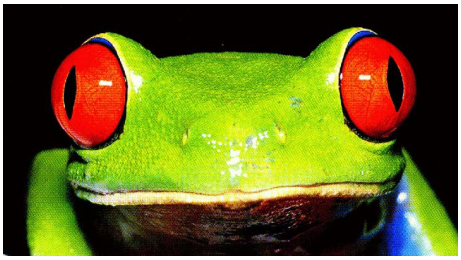


Figure 1.1: The eyes are the most important senses for a frog.

1982] and states the scientific field where the thesis at hand is placed.

Why is it worth doing research on computer vision? The answer to this question can be split up into two parts: First, from a researcher's perspective and due to the human pioneering spirit, *i.e.*, simply because we want to find out if we can do it. Second, because there exist a huge amount of potential applications. Some of these are optical character recognition (OCR), web-based image search, industrial machine inspection, biomedical imaging, surveillance, 3D model building and photogrammetry, automotive safety, biometrics, robotics, *etc.*. See also David Lowe's website of industrial vision applications [1]. Overall, computer vision already now comprises a multi-billion dollar market with expected steady growth.

Vision is hard The human visual system rapidly and effortlessly recognizes a large number of diverse objects despite large variations in the object's position, pose, lighting and background clutter. Additionally, we can easily segment an object, analyze its shape and track it. Building computational systems that are able to achieve the same performance is extremely hard. One of the reasons for this difficulty is that "reverse engineering the brain" in order to emulate it on machines is very complicated. Although cognitive science has made large progress in understanding the brain's solution to perform visual tasks (see above), we are still far from fully understanding how human perception works. Additionally, unlike humans, we provide computers with digital data and from a machine's perspective, an image is nothing else than a matrix of numbers. The size and the quality of the matrices may vary a lot. Hence, the way how computer vision is done today can be also interpreted as searching for useful information in matrices and researchers still argue if this is the right path to follow.

Computer vision is an inverse discipline, that is we have to find a solution for a problem where we get provided by an insufficient amount of information; and inverse problems are typically ill-posed, *i.e.*, there does not a exist a unique solution [Hadamard,

[1] http:://www.cs.ubc.ca/spider/lowe/vision.html (01.04.2010)

1902]. Hence, computer vision has often to make use of probabilistic approaches and the success of the methods relies on how good our probabilistic models are and on the quality of information with which we feed them. Finally, the human visual system is able to cope with large scales of data and thus acts as a massively parallel computer, comprised of billions of elements. Hence, when doing vision on machines we also have to cope with a huge amount of data, which requires both large computational power and highly efficient algorithms in order to deliver results in acceptable time if not real-time. Altogether, these are big challenges both for algorithm designers and hardware developers.

The role of Machine Learning and Training Samples Computer vision can be subdivided into several disciplines, such as, structure from motion, segmentation, reconstruction, action recognition, *etc.* [Szeliski, 2010]. In this thesis, we are mainly dealing with visual recognition and classification as well as object tracking. A typical machine perception or pattern recognition system can be subdivided into the following steps (See also Duda [Duda et al., 2001]):

1. Sensing
2. Segmentation
3. Feature extraction and selection
4. Classification
5. Post-processing

The success of a recognition system clearly depends on the quality of either of these steps; however, especially good representations in terms of features and training accurate classifiers mostly determine the overall quality of a recognition system. Hence, in recent years, further developments in terms of representation and novel machine learning algorithms have brought the highest accuracy improvements. In fact, especially machine learning techniques become increasingly relevant for computer vision and, according to this observation, this is also the reason why in this thesis we mainly focus on the development and application of learning and classification algorithms for computer vision.

The task of a machine learning algorithm is to, based on provided training samples, train classifiers that predict the labels of samples that have not been observed during training. In practice, both the training samples and their corresponding labels are provided by a human labeler. The learners are thus called supervised methods. A lot of research has been focused on developing new classifiers and learning algorithms. If enough and proper training samples exist, these approaches can obtain very high recognition and classification performances. However, one fact that has been often ignored or overlooked in

the literature is that for many practical problems obtaining enough labeled data is a very tedious, time-consuming and costly task and is sometimes not even possible. Additionally, the critical labeling process is often subject to human errors which deteriorates the classification performance.

The lack of sufficient labeled training data and the problems involved in the labeling process have led to recent attentions towards the investigation of unsupervised and semi-supervised learning (SSL) methods. Unsupervised learning methods try to find an interesting (natural) structure in the data using training samples without their corresponding labels, *i.e.*, unlabeled data. Although unsupervised learning in principle is convenient because human labeling can be fully avoided, it can by far not reach the discriminative performance of supervised learning algorithms. In contrast, *semi-supervised learning* tries to exploit both, a reasonably small amount of labeled data and a large amount of unlabeled data. It thus tries to combine the benefits of both supervised learning – because highly discriminative classifiers are delivered – and unsupervised learning – because it holds the potential to reasonably exploit a massive amount of unlabeled data.

Especially computer vision motivates the research on semi-supervised learning algorithms because the world wide web and the mass production of digital cameras at very low costs let the number of digital images and videos increase in vertiginous numbers. It is clear that one wants to exploit this data in order to get better, for instance, categorization and recognition systems. Since some of the data are labeled and many are unlabeled or ambiguously labeled, *e.g.*, due to surrounding text, semi-supervised learning algorithms are the natural choice for exploiting the huge amount of digital images.

Finally, what makes semi-supervised learning also interesting for computer vision is that humans use the same learning strategy. In other words, there exists a large agreement among experts that the power of human perception is also a result of a long-winded learning process where we continuously observe a considerable exorbitant number of data, yet, most of it unlabeled. Hence, semi-supervised learning is not only useful from a technical or machine learning perspective it is also biologically plausible.

Off-line versus On-line Learning Most of the previously proposed semi-supervised learning algorithms are so called off-line methods. This means that they get the entire training set, both labeled and unlabeled, at the same time. Off-line learning eases optimization because the entire data can be exploited at once and usually delivers good results. Moreover, training and testing of the classifier are clearly separated. As we previously stated, semi-supervised learning is natural, however, doing it off-line is not natural at all. In the real world, learning is usually a continuous process. This means that if we want to reflect biological systems on machines, we also need incremental or on-line learning methods and it is thus worthwhile to investigate on-line semi-supervised learn-

ing algorithms. Additionally, in practice there exist many scenarios where the data arrives sequentially and systems have to be updated without re-training the entire model. This is, for example, the case in robot navigation or visual object tracking. Finally, in order to learn from large amounts of data, as argued in the SSL literature, it is essential to apply algorithms that are computationally efficient and can operate with limited amounts of memory. On-line learning methods are faster and require less amount of memory compared to off-line algorithms which makes them ideal for semi-supervised learning tasks.

Object Tracking Semi-supervised learning methods are mainly applied in order to increase the accuracy of a classifier by exploiting additional huge amounts of unlabeled data. However, SSL can also be applied on a task which, at the first glance, does not seem to be related to learning from both labeled and unlabeled data, *i.e.*, visual object tracking. In fact, if we consider a tracking task, where an a priori unknown object is marked in the first frame a powerful tracking approach is to learn a binary classifier with the marked patch as positive sample and the surrounding patches as negatives, respectively. Then, the tracking of the object is performed by using this classifier to re-detect the marked object in the subsequent frames. Hence, such approaches are also called *tracking-by-detection* methods. In order to make the classifiers adaptive towards rapid appearance and illumination changes, typically the classifier updates itself on the re-detected object, according to hand-designed update rules. As already mentioned, such approaches yield highly accurate trackers which are also fast because the object has only to be discriminated versus its local background and the model complexity can be kept very low. Nevertheless, one problem that all of these methods have in common is that slight errors during the self-updating process can easily accumulate and finally lead to failure of tracking, *i.e.*, the object is lost. This is also called drifting. One reason for drifting of these tracking-by-detection approaches lies in the fact that supervised learners are abused for an in principle unsupervised learning task. In fact, one can easily see that labeled data is only available at the first frame when the object is marked. In all subsequent frames the task of the tracker is to act autonomously and without getting any additional labels for the data it is observing. Hence, the learner over time has to be able to exploit both labeled data and unlabeled data, which is a natural semi-supervised learning problem. Following this observation, in this thesis, we will propose novel on-line semi-supervised learning algorithms and we will show that applying these methods to visual object tracking results in more robust tracking results.

1.1 Contribution

The content of this thesis is mainly based on the work presented in [Leistner et al., 2008, Grabner et al., 2008, Leistner et al., 2009b, Leistner et al., 2009a, Saffari et al., 2009b] and some further work that at this time is currently under reviewing process for major conferences and journals, respectively. Overall, this work is the result of strong collaborations, mostly with Helmut Grabner, Amir Saffari, Peter M. Roth and Prof. Horst Bischof, and others. The main contributions of this thesis can be summarized as follows:

In general, we introduce novel semi-supervised ensemble learning methods and show their suitability for computer vision problems. We concentrate on ensemble methods such as boosting and random forests because they are able to deliver highly accurate results and are frequently used in computer vision. In particular, as first contribution, we present an algorithm that combines semi-supervised boosting and visual similarity learning. The motivation for such a combination stems from the fact that semi-supervised learning methods often demand a priori given similarity or distance functions in order to guide the learning algorithm during exploitation of the unlabeled data. However, especially in computer vision, it is often difficult to decide which similarity measure should be taken. We take the limited amount of labeled data in order to train a discriminative pair-wise similarity – pair-wise learning has further the advantage that less labeled data is needed – and use this similarity for semi-supervised boosting on visual data. Furthermore, we show that in principle any classifier can be used as prior and the classifiers can be combined in a principled way. In the experiments, we show that our approach makes sense for computer vision and demonstrate several applications, such as training object detectors from a small amount of labeled data or transferring classifier knowledge between different camera views.

The second main contribution of this thesis studies the development of novel *on-line* semi-supervised boosting methods and their application to the task of visual object tracking. In more detail, we show how on-line boosting can be extended to semi-supervised learning and propose a novel algorithm called *On-line SemiBoost*. Based on this extension, we formulate tracking as an one-shot semi-supervised learning task and demonstrate that our semi-supervised boosting algorithms lead to much more stable tracking results. Additionally, we study the robustness of on-line boosting methods which are usually highly susceptible to class-label noise. As a further contribution, we present an on-line boosting algorithm which allows for the flexible incorporation of robust loss functions and extend this algorithm also to semi-supervised learning.

Besides boosting, random forests are currently one of the most frequently applied ensemble methods in computer vision. Since random forests usually demand a huge amount of labeled data in order to enroll their full potential, in this thesis, we also propose an algorithm that leverages random forests for semi-supervised learning. This algorithm

has several benefits over previously proposed SSL methods: first, it is inherently multi-class. Second, it is fast and scales well to large amounts of data and, third, it provides an inherent mechanism to detect during the learning if unlabeled data helps to improve the classification accuracy or not. We further show how to extend random forests to on-line learning and, as with boosting, show how to perform on-line SSL with random forests.

Finally, we also study multiple instance learning because it is an increasingly popular machine learning paradigm and has been recently applied to numerous computer vision applications, such as, object detection, content-based image retrieval and tracking. We will show that multiple instance learning is very similar to semi-supervised learning and present a method that allows for multiple instance learning with random forests.

1.2 Outline

The thesis is organized as follows: First, in Chapter 2 we review the main theoretical issues about statistical learning basics as well as ensemble methods that are used throughout this thesis. In the following Chapter 3, we introduce the concept of semi-supervised learning and give a more detailed overview of the most important approaches and related work. In Chapter 4, we present a semi-supervised boosting method (*i.e.*, SemiBoost) that uses learned similarity functions. We illustrate the efficiency of the method on tasks such as car and face detection. In general, also for the remaining chapters, we will give experimental results inside a chapter if it is necessary to illustrate the behavior of a method. Mostly, we will do this for the off-line methods. All on-line methods will be evaluated on the tracking task in Chapter 9.

In Chapter 5 we introduce our on-line semi-supervised boosting algorithm. In the chapter, we also study the robustness of on-line boosting towards class label noise and present an on-line algorithm which is less susceptible to noise. Based on this algorithm, we will then also introduce a more robust on-line semi-supervised boosting method.

Chapter 6 demonstrates how one can perform semi-supervised learning with another popular ensemble method, *i.e.*, random forests, and in Chapter 7 we will also present an on-line version of this approach. Chapter 8 will discuss multiple instance learning (MIL) as another popular strand of machine learning that is very similar to SSL, and we will introduce a novel MIL algorithm based on random forests.

In Chapter 9, we first show how object tracking can be formulated as an on-shot semi-supervised learning task, using the methods proposed in this work, and we will discuss several issues that arise in this context. In the tracking experiments, we analyze most of the on-line methods presented in this thesis on benchmark tracking sequences. The thesis concludes with the final Chapter 10 where we summarize the work and give an outlook to potential future work.

Chapter 2

Preliminaries and Notations

The goal of this thesis is to investigate and develop new semi-supervised learning methods. This chapter reviews some basic machine learning notations and learning methods that are frequently used throughout the thesis.

2.1 Machine Learning and Classification

Machine learning is a major sub-field of artificial or computational intelligence and is concerned with the study of algorithms that improve through experience [Mitchell, 1997, Alpaydin, 2010]. Formally, one deals with a labeled dataset $\mathcal{D}^L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{|\mathcal{D}^L|}, y_{|\mathcal{D}^L|})\} \subseteq \mathcal{X} \times \mathcal{Y}$ where $\mathbf{x}_i \in \mathcal{X} = \mathcal{R}^d$ and $y_i \in \mathcal{Y}$. Note that as notations, usually lower case letters (*e.g.*, x, λ) are used for scalar variables and bold face letters (*e.g.*, $\mathbf{x}$, $\boldsymbol{\lambda}$) denote vectors. Sets are represented by calligraphy letters (*e.g.*, $\mathcal{X}$). For a binary classification problem, usually $\mathcal{Y} = \{+1, -1\}$ and the samples are split into two sets $\mathcal{X}^L = \mathcal{X}^+ \cup \mathcal{X}^-$ of all samples with a positive class and the set of all samples with negative class, respectively. Then, a classification function or hypothesis in form of $H : \mathcal{X} \to \mathcal{Y}$ is trained using the labeled samples. Note that we use the letters H, F and G interchangeably to denote classification functions. Methods that are provided with both data and their corresponding labels are also called *supervised learning* methods. The goal of supervised learning is to provide a classifier which has low prediction error on future data that has not been observed during training. In more detail, for a given label $y \in \mathcal{Y}$ and if the misclassification error is $H(X) \neq y$, we can define the *generalization error* (GE) to be

$$GE(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\mathbb{I}(H(x) \neq y)], \tag{2.1}$$

where $\mathcal{D}$ is a joint distribution over $\mathcal{X} \times \mathcal{Y}$, $\mathbb{I}(\cdot)$ is the indicator function and θ is a parameter vector. $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}$ is the expected prediction over an independent test sample. $\mathbb{I}(\cdot)$ is

also called $0-1$ loss function. In general, a loss function $\ell(\mathbf{x}, y, h(x)) \in [0, \infty]$ measures the empirical quality of the hypothesis $h(\mathbf{x})$ on the example $(\mathbf{x}, y)$. Further, $\mathcal{H}$ is the set of all possible hypotheses and $\ell(\cdot)$ is a function which maps to real-valued predictions.

During learning, the goal is to find hypotheses and values for θ that minimize the generalization error. However, this is usually a difficult task since $\mathcal{D}$ is not known and one can only minimize GE on the trainingset $\mathcal{D}^L$. Furthermore, directly minimizing the $0-1$ loss is also difficult since it is not *convex*. So, we use a convex upper bound of the $0-1$ loss which is easier to minimize and the training error becomes

$$\hat{GE}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i; \theta). \tag{2.2}$$

$\hat{GE}(\theta)$ is also known as *empirical risk* and for $\ell(\cdot)$ one usually takes a convex function such as $e^{-(\mathbf{x}_i, y_i; \theta)}$. Note that the empirical risk is only a surrogate for the real generalization. Two additional import terms in machine learning that help to understand the success and failure of classification methods are *bias* and *variance* [Friedman et al., 2001]. In particular, we can measure the mean-squared-error of the learned classifier $H(\mathbf{x})$ to the true conditional probability $p(y|\mathbf{x})$ as $err\big(H(\mathbf{x}), p(y|\mathbf{x})\big) = \mathbb{E}[(H(\mathbf{x}) - p(y|\mathbf{x}))^2]$ [Alpaydin, 2010]. This term we can further split up to

$$err\big(H(\mathbf{x}), p(y|\mathbf{x})\big) = \underbrace{\mathbb{E}[(H(\mathbf{x}) - \mathbb{E}[H(\mathbf{x})])^2]}_{variance} + \underbrace{(\mathbb{E}[H(\mathbf{x})] - p(y|\mathbf{x}))^2}_{bias}. \tag{2.3}$$

The variance is the variation of the prediction of the learned classifiers, *i.e.*, the squared difference between the predictions on the training samples and its average prediction. The bias measures how much the expected predictions vary from the correct ones. Reliable learning methods should have both low variance and low bias; however, since this is difficult to achieve – because low bias often results in higher variance and vice versa – one has often has to make a so called "bias-variance trade-off".

Optimizing $\hat{GE}(\theta)$ also may contain the risk of *overfitting*, *i.e.*, the classifier will fit perfectly to the statistical noise given by the training data but will poorly generalize. A popular remedy for overfitting is *regularization*, which penalizes complex solutions for θ. In particular, in regularization, one extends $\hat{GE}(\theta)$ with an additional regularization term $r(\theta)$ so that it becomes

$$\hat{GE}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i; \theta) + r(\theta). \tag{2.4}$$

In order to get an unbiased measure of GE we can also split up labeled samples $\{(\mathbf{x}_i, y_i)\}_{J=n+1}^{n+m}$ from the training set $\mathcal{D}^L$ and measure the *test error* on these samples.

In supervised learning, one can further differentiate between *generative* and *discriminative* methods. Generative methods try to capture the class conditional density $p(\mathbf{x}|y)$. For classification one can then apply Bayes theorem [Duda et al., 2001]:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \tag{2.5}$$

Examples for popular generative models are principal component analysis (PCA), independent component analysis (ICA), latent Dirichlet allocation (LDA), Gaussian Mixture Models (GMM) or Hidden Marcov Models (HMM) (See [Duda et al., 2001] for details on these approaches). Discriminative models compute $p(y|\mathbf{x})$ directly and thus are mainly focused on how well they separate positives from negatives, for a two-class problem. For the classification task, discriminative methods are mostly preferred because in practice they have been shown to be more effective. Popular discriminative learning approaches are, for instance, boosting [Freund and Schapire, 1999] or support vector machines (SVMs) [Schoelkopf and Smola, 2002].

In contrast to supervised approaches, *unsupervised methods* do not have labels $\mathcal{Y}$ for the data and aim to find an interesting (natural) structure in $\mathcal{X}$ using only unlabeled data $\mathcal{D}^U = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{D}^U|}\} \subseteq \mathcal{X}$. Since there are no labeled samples available, unsupervised learning can also be considered as estimators for the density $p(\mathbf{x})$ which is likely to have generated the underlying distribution $\mathcal{X}$. Popular applications of unsupervised methods are, for instance, clustering or dimensionality reduction [Hinton and Sejnowski, 1999, Duda et al., 2001].

2.2 Off-line versus On-line Learning

We have seen that the goal of machine learning is to train predictor functions on provided training data. We now can further discriminate among different learning methods depending on *how* the data and in particular in *which order* they are provided to the learning algorithm. Usual learning methods are called *batch* or *off-line* learning methods. In these approaches, all the training data is given in advance, *i.e.*, the learning algorithm can observe all samples simultaneously. In order to learn a model, these methods repeatedly process the entire training set until a certain stopping criterion is met; for instance, the training error has fallen under a certain threshold or the maximum number of iterations has been reached.

In contrast to off-line methods, there exist *incremental* or *on-line* approaches where the data is usually not provided at once but arrives sequentially. Both incremental and on-line methods process only one sample at a time and then update the model. The main difference between the two is that in on-line learning each sample is discarded after an

update, in incremental learning not. On-line learning is always incremental, whereas, incremental learning can be done either on-line or off-line[1]. Formally, for any given sequence of training samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ a sequence of hypotheses $h_1, \ldots, h_T$ is generated, such that h_t only depends on h_{t-1} and the current sample $(\mathbf{x}_t, y_t)$. A training sample can be kept or can be discarded after having updated the model but usually no re-processing of previous samples is necessary. In contrast to off-line learning, training and testing the classifier can take place at any time and are not fully separated.

On-line learning takes place in consecutive rounds and can also be explained in form of a "question-answer" game between the learner and an expert or adversary environment, where a "question-answer pair" corresponds to a training sample. Suppose you have a learner that predicts the shortest daily path from your home to your working place if you travel by car. The classifier should be updated on a daily basis, should be adaptive, *e.g.*, to temporal construction sites, and you want predictions to be available all the time. The decision of this problem depends on several inputs, such as distance, allowed travel speed and traffic-jam probability, *etc.*. These inputs are selected by the expert and provided to the learner. To answer the question for the shortest path, for each training sample $\mathbf{x}_t$ the learner builds a hypothesis which maps the set of questions to a prediction $\hat{y}_t$. Then, the expert reveals the true label y_t. The quality of the learner's answer is assessed by a loss function that measures the discrepancy between the predicted answer and the correct one. The goal of the learner is to minimize the cumulative loss suffered along this run. To achieve this goal, the learner updates the hypothesis after each round, *i.e.*, after each daily trip to work, so as to be more accurate in later rounds. Hence, by deducing information from previously observed samples, the learning algorithm tries to improve its predictions from day to day. The example again illustrates another substantial characteristic of on-line models: an applicable classifier is already present after having observed the first training sample, and training and evaluation phases are less separated than in off-line methods.

One important notion in on-line learning is the *regret factor*. Given any fixed hypothesis $h \in \mathcal{H}$, the regret of an on-line learning algorithm is defined as

$$R(h, T) = \sum_{t=1}^{T} \ell(h_t, (\mathbf{x}_t, y_t)) - \sum_{t=1}^{T} \ell(h, (x_t, y_t)), \tag{2.6}$$

where T is the number of hypotheses. As we can see, the regret is the difference of the cumulative loss of the on-line learner and the cumulative loss of the optimal fixed hypothesis h. Due to its higher practical relevance, on-line learners are often analyzed in terms of their regret. Note that throughout this thesis we use the term batch-learning and off-line as well as incremental- and on-line learning interchangeably. For further details

[1] Definition by Warren S. Sarle, "ftp://ftp.sas.com/pub/neural/FAQ2", (22.4.2010)

and general discussions about on-line learning methods we refer the reader to [Blum, 1996, Shalev-Shwartz, 2007] and the references therein.

2.3 Ensemble Methods

There exist a reasonable amount of different machine learning methods. Most of them perform more or less different depending on the tasks and how their hyperparameters are tuned for a given problem. This is also theoretically underpinned by the *No Free Lunch Theorem* [Wolpert, 1996], which states that there does not exist a single learning algorithm that in any domain always induces the most accurate learner. Hence, it often makes sense to combine multiple learners to an ensemble, which performs better than each individual method alone. In particular, ensemble methods are learning algorithms that are a combination of several learning methods, *i.e.*, base learners [Dietterich, 1998], and are thus also called *meta learning* approaches because they work on top of other learning algorithms. The combination of the different learners can be by weighted or unweighted voting. The underlying idea of ensemble methods is to find a combination of the base learners that overall performs better than the individual parts. Typical conditions on the base learners are that they are both accurate and diverse. In the terminology of ensemble methods a classifier is accurate if it has a generalization error lower one half; *i.e.*, it performs better than random guessing. The classifier can be considered as diverse if they make different errors on new data points $\mathbf{x}$. Consider the case of building an ensemble of n classifiers $H = \{h_1, h_2, \ldots, h_n\}$ with $n \geq 3$. If the error rates of the n hypotheses are all smaller $\frac{1}{2}$ and if the errors are independent, then the probability that the majority vote will be wrong can be illustrated by the area under a binomial distribution where more than $\frac{N}{2}$ base classifiers are wrong. According to this binomial distribution, [Dietterich, 1998] shows that for an ensemble of, *e.g.*, 21 classifiers with each having an error of $\frac{1}{3}$ the probability that the ensemble is wrong is 0.026, which illustrates the power of combining several classifiers to an ensemble.

One of the simplest forms of ensembles are based on *Bayesian Voting*. If we have several hypotheses $h \in \mathcal{H}$, Bayesian Voting simply combines all hypotheses by building their weighted sum in form of

$$P(f(x) = y|S, \mathbf{x}) \sum_{h \in \mathcal{H}} h(\mathbf{x}) P(h|S), \tag{2.7}$$

where S is a training sample. As can be seen, the individual hypotheses determine their weights by themselves with their posterior probability $P(y|\mathbf{x})$.

Besides Bayesian Voting, *Stacking* is another simple and popular ensemble method [Wolpert, 1992]. There, the underlying idea is to split the training data into a so called *held-in* and

held-out dataset, respectively. In a cross-correlation manner, the individual learning methods are trained on the held-in set and their weighted combination is determined depending on their performance on the held-out set.

Another popular ensemble method is called *Bagging* or Bootstrap aggregating [Breiman, 1996a]. In Bagging, an ensemble is built by letting several classifiers sub-sample from the original training set with replacement. As a result, each classifier on average sees $1 - \frac{1}{e}$ of the data, which corresponds to $\approx 63\%$. Thereby, for each learner slightly different "views" on the learning problem are generated which both increases the diversity among the learners and eases the learning tasks for the base classifiers because they can concentrate on fewer samples. The final classifier is simply built by the non-weighted combination of the base classifiers in form of

$$F(\mathbf{x}) = \sum_{t=1}^{T} f_t(\mathbf{x}). \tag{2.8}$$

As Breiman showed, Bagging is a very successful method to reduce the variance of learning functions. In the following, we will review a method that was developed out of bagging and is able to minimize both bias and variance.

2.3.1 Boosting

Boosting [Freund and Schapire, 1999] is a general method for improving the accuracy of any given learning algorithm. It is a typical ensemble method, *i.e.*, the algorithm combines several weak classifiers to a strong one in the form

$$F(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}), \tag{2.9}$$

where α_t determines the influence of the t^{th} weak learner. In contrast to previous ensemble methods, *e.g.*, bagging, boosting forms the ensemble in an iterative process, *i.e.*, the committee evolves over time, where in each iteration t one weak classifier is added to the ensemble until a certain stopping criterion is met. Boosting has shown to converge quadratically if the error of each weak learner is less than 50%; *i.e.*, the weak learner is at least better than random guessing.

Although [Kearns and Valiant, 1994] were the first to mention that several weak learners can be "boosted" to a strong one, it was the introduction of *AdaBoost* by Freund and Schapire [Freund and Schapire, 1997] that finally led to the popular boosting variant as we know it today and helped it to become one of the most powerful learning algorithms. AdaBoost, during learning, keeps a weight distribution $D_t(i)$ over the training samples. According to this distribution, in each iteration boosting selects the best weak learner

and adds it to the model. After each iteration t the samples are re-weighted according to a given loss function, *e.g.*, $e^{-yF(x)}$ in case of AdaBoost, which forces the algorithm to concentrate on hard samples and leave out easy samples, respectively. We illustrate the method in Algorithm 16. See also [Freund and Schapire, 1999] for a good introduction.

Algorithm 2.1 AdaBoost [Freund and Schapire, 1997]

Require: Training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ with $y_i \in \mathcal{Y} = \{-1, +1\}$
Set $D_1(i) = \frac{1}{m}$
Require: max iterations T
for $t = 1, 2, \ldots, T$ **do**
    // Normalize weights w_i so that D_t is a probability distribution
    $w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{m} w_{t,j}}$
    Train weak learners using D_t
    Get weak hypothesis $fWeak_t : \mathcal{X} \to \{-1, +1\}$ with error
    $\epsilon_t = Prob_{i \sim D_i}[fWeak_t(\mathbf{x}_i) \neq y_i]$
    if $\epsilon_t = 0 or \epsilon_t > \frac{1}{2}$ **then**
        break
    end if
    Set $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
    // Update weight distribution

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if fWeak_t(\mathbf{x}_i) = y_i \\ e^{\alpha_t} & fWeak_t(\mathbf{x}_i) \neq y_i \end{cases} = \frac{D_t(i) e^{-\alpha_t fWeak_t(\mathbf{x}_i)}}{Z_t},$$

    where Z_t is a normalization factor
end for
Output the final hypothesis: $fStrong = sign\Big(\sum_{t=1}^{T} \alpha_t \cdot fWeak_t(\mathbf{x})\Big)$

Friedman *et al.* [Friedman et al., 2000] showed that boosting approximates adaptive logistic regression, which explains the power of the method. They also showed in their "statistical view of boosting" that the output of a boosted classifier $F(\mathbf{x})$ can be transformed via a monotone logistic transformation into a probability in form of

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-F(\mathbf{x})}}. \tag{2.10}$$

As was already mentioned, boosting is very popular, and besides the fact that it is a very powerful learning algorithm its popularity also stems from the fact that it is rather easy to implement. Another important property of the algorithm that led to its overwhelming success is the fact that virtually no parameter tuning is necessary. In fact, the

only parameter that has to be set is the maximum number of iterations T. One may argue that the setting of this parameter is crucial because the algorithm might tend to overfit if T is set to a large number. However, as several researchers have shown [Rudin et al., 1999], boosting does hardly overfit, even if T is very large. Furthermore, boosting still decreases the generalization error, even long after the training error has reached zero. Theoretically, [Freund and Schapire, 1999] showed that this can be explained by the fact that boosting still increases the margin over the training samples after the training set is perfectly separated which further decreases the generalization error.

2.4 On-line Boosting

The AdaBoost algorithm as discussed above is an off-line learning method, *i.e.*, it assumes having access to the entire training data space at once. However, there exist also a popular extension of boosting towards on-line learning, which is used in many application and is shortly discussed in this section.

As we have seen, boosting additively combines several weak classifiers $f_t(\mathbf{x})$ to a strong one $F(\mathbf{x})$. During learning, boosting keeps a weight distribution over the training samples in order to concentrate on hard samples and to reduce the effort on easy-to-learn samples, respectively. Since in the on-line case the training data arrives sequentially and samples are usually not stored[1], the difficult task is to estimate the weight distribution over the training samples. Oza and Russel [Oza and Russell, 2001, Oza, 2001] introduced an on-line version of AdaBoost. The algorithm starts with an ensemble of T weak classifiers. [Oza and Russell, 2001] then proposed to estimate the weight distribution of the sequentially arriving training samples using a Poisson distribution and compute the importance λ_i of the i^{th} sample by propagating it through the set of weak classifiers. The importance plays the role of the weight distribution $p(\mathbf{x_i})$ in the off-line case. In fact, λ is increased proportional to the error e of the weak classifier if the sample is still misclassified and decreased, otherwise. The error of the weak classifier $\hat{e} = \frac{\lambda^w}{\lambda^w + \lambda^c}$ is estimated by the sum of correctly λ^c and incorrectly λ^w importances seen so far. Oza has proven that the approach converges to its off-line counterpart in case of using Naïve Bayes classifiers. Note that for arbitrary weak classifiers a proof of convergence is still an open research problem. However, the work of Oza suggests that if the used on-line weak learners are converging to their off-line counterparts also the entire on-line boosting algorithm will converge.

[1] Note that there in principle does not exist a law that forbids the storage of training samples for the on-line setting. In this work, however, we refer to the term "*on-line*" to learning classifiers without the necessity of storing samples.

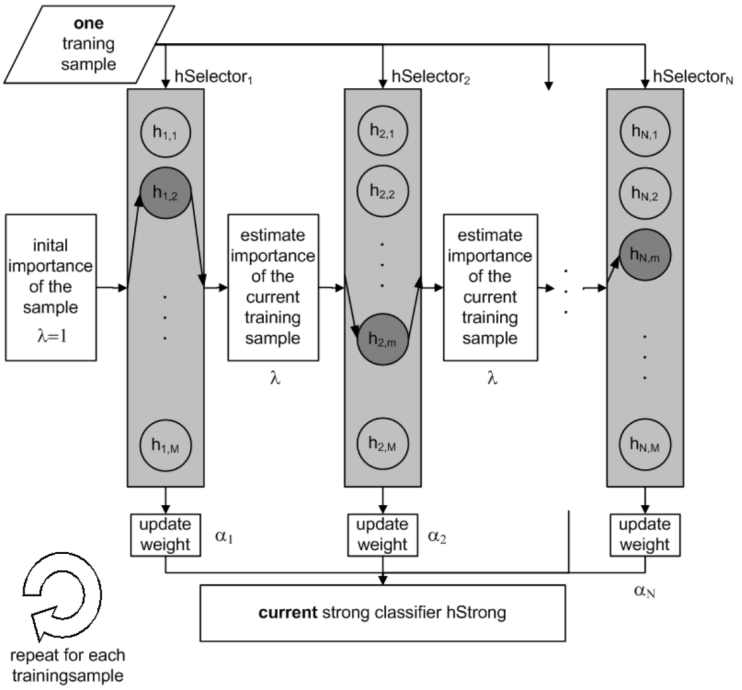


Figure 2.1: On-line boosting for feature selection [Grabner and Bischof, 2006] .

2.4.1 On-line Boosting for Feature Selection

In off-line learning, boosting is often used for feature selection [Tieu and Viola, 2000], *i.e.*, the t^{th} weak corresponds to feature f_t drawn from a feature pool $\mathcal{F}$ which has the lowest weighted training error. This has been shown to be particularly useful in computer vision applications, where features can be simple Haar-like filters [Viola and Jones, 2001, Papageorgiou et al., 1998] or keypoints. Since feature selection is also demanded in the on-line setting, Grabner and Bischof [Grabner and Bischof, 2006] introduced on-line boosting for feature selection.

In their method, the main idea is to introduce selectors $\mathcal{S} = h^{sel}(\mathbf{x})$, where each selector holds M weak classifiers $\{h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})\}$. Boosting is then performed not directly on the weak classifiers but on the selectors. When training a selector, its M weak classifiers are trained and the one with the lowest estimated error is selected, *i.e.*, $h^{sel}(\mathbf{x}) = \arg\min_m e\left(h_m(\mathbf{x})\right)$.

The workflow of the AdaBoost on-line training framework used for feature selection is as follows: A fixed number of N selectors $h_1^{sel}, .., h_N^{sel}$ are initialized with random features. The selectors are updated, as soon as a new training sample $\langle \mathbf{x}, y \rangle$ is available, and the weak classifier with the smallest estimated error is selected. For the updating process of the weak classifier any on-line learning algorithm is applicable. Finally, the weight α_n of

the n-th selector h_n^{sel} is updated and the importance λ_n is passed to the next selector h_{n+1}^{sel}. Contrary to the off-line version, the on-line classifier is available at any time of the training process as a linear combination of the N selectors. While the method's schematics are illustrated in Figure 2.1 the detailed steps are depicted in Algorithm 2.2.

Algorithm 2.2 On-line Boosting for feature selection

Require: training (labeled or unlabeled) example $\langle \mathbf{x}, y \rangle$, $y \in \{-1, +1, \#\}$
Require: strong classifier F (initialized randomly)
Require: weights $\lambda_{n,m}^{corr}$, $\lambda_{n,m}^{wrong}$ (initialized with 1)

// for all selectors h_n^{sel}
for $n = 1, 2, \ldots, N$ **do**
    // update the selector h_n^{sel}
    for $m = 1, 2, \ldots, M$ **do**
        // update each weak classifier
        $f_{n,m} = \mathrm{update}(h_{n,m}, \langle \mathbf{x}, y \rangle, \lambda)$
        // estimate errors
        if $f_{n,m}^{weak}(\mathbf{x}) = y$ **then**
            $\lambda_{n,m}^{corr} = \lambda_{n,m}^{corr} + \lambda_n$
        else
            $\lambda_{n,m}^{wrong} = \lambda_{n,m}^{wrong} + \lambda_n$
        end if
        $e_{n,m} = \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{correct} + \lambda_{n,m}^{wrong}}$
    end for
    // choose weak classifier with the lowest error
    $m^+ = \arg\min_m(e_{n,m})$, $e_n = e_{n,m^+}$, $f_n^{sel} = f_{n,m^+}$
    if $e_n = 0$ or $e_n > \frac{1}{2}$ **then**
        exit
    end if
    // calculate voting weight
    $\alpha_n = \frac{1}{2} \cdot \ln\left(\frac{1-e_n}{e_n}\right)$
    // update importance weight
    if $f_n^{sel}(\mathbf{x}) = y$ **then**
        $\lambda = \lambda \cdot \frac{1}{2 \cdot (1-e_n)}$
    else
        $\lambda = \lambda \cdot \frac{1}{2 \cdot e_n}$
    end if
    // replace worst weak classifier with a new one
    $m^- = \arg\max_m(e_{n,m})$
    $\lambda_{n,m^-}^{corr} = 1$; $\lambda_{n,m^-}^{wrong} = 1$;
    get new f_{n,m^-}^{weak}
end for

2.4.2 Random Forests

As we have seen above, boosting is a further development of bagging and usually outperforms bagging on many problems. In turn, Random Forests (RF) are closer to the original bagging idea. Random Forests were proposed by Breiman [Breiman, 2001] and are bagged ensembles of N de-correlated decision trees . Trees are perfect candidates for bagging because they are usually high-variance as well as low-bias learners.

The main idea of random forests is to reduce the correlation among the trees in order to improve the variance reduction of bagging by growing trees that perform random selection on the input variables. We thus talk about *randomized trees*. We denote the n^{th} tree of the ensemble as $f_n(\mathbf{x}) = f(\mathbf{x}, \theta_n) : \mathcal{X} \rightarrow \mathcal{Y}$, where θ_n is a random vector capturing the various stochastic elements of the tree, such as the randomly sub-sampled training set or selected split tests at its decision nodes. We also denote the entire forest as $\mathcal{F} = \{f_1, \cdots, f_N\}$, where N is the number of trees in the forest. During training of randomized trees each decision node of the tree creates a set of random tests $m \leq p$, where p is the number of all input variables. m is usually set to $\sqrt{p}$. Each node selects the best split according to some quality measurement which scores the potential information gain

$$\Delta H = -\frac{|I_l|}{|I_l| + |I_r|} H(I_l) - \frac{|I_r|}{|I_l| + |I_r|} H(I_r), \tag{2.11}$$

where I_l and I_r are the left and right subsets of the training data, respectively. The node score $H(I)$ is usually measured using the entropy $H(I) = -\sum_{i=1}^{K} p_i^j \log(p_i^j)$ or the Gini $H(I) = -\sum_{i=1}^{K} p_i^j (1 - p_i^j)$, where p_i^j is the label density of class i in node j. The recursive training continues until a maximum depth is reached or no further information gain is possible. The trees are usually grown to their full size without pruning. We depict the training process in Algorithm 2.3.

Each tree in the forest is built and tested independently from other trees, hence the overall training and testing procedures can be performed in parallel. During the training, each tree receives a new bootstrapped training set generated from the original training set by sub-sampling with replacement. We refer to those samples which are not included during the training of a tree as the *Out-Of-Bag* (OOB) samples of that tree. These samples can be used to compute the so called *Out-Of-Bag-Error* (OOBE) of the tree as well as for the ensemble. According to Breiman [Breiman, 1996b], the OOBE is an unbiased estimate of the generalization error and one can thus use it in order to get better estimates of model parameters. This is very similar to performing n-fold cross-validation; however, in random forests it is part of the learning strategy and is thus inherently provided.

Random forests are multi-class classifiers and we can write the estimated probability

Algorithm 2.3 Random Forest [Breiman, 2001]

Require: Training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ with $y_i \in \mathcal{Y} = \{-1, +1\}$
Require: The size of the forest N
Require: Maximum depth D of the decision trees
1: **for** $n = 1, 2, \ldots, N$ **do**
2: Bootsrap with replacement training samples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ from $\mathcal{X} \times \mathcal{Y}$
3: Train a random-tree via recursively repeating the following steps:
4: **while** depth $d < D$ and there exist non-pure nodes **do**
i Randomly select m variables from the p input variables
ii For each selected variable calculate the potential info gain and pick the best
iii Split the node and add two child nodes
5: **end while**
6: **end for**
7: Output the final forest $\mathcal{F}$

for predicting class k for a sample as

$$p(k|\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} p_n(k|\mathbf{x}), \tag{2.12}$$

where $p_n(k|\mathbf{x})$ is the estimated density of class labels of the leaf of the n^{th} tree. The final multi-class decision function of the forest is defined as

$$C(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} p(k|\mathbf{x}). \tag{2.13}$$

Breiman [Breiman, 2001] defined the classification *margin* of a labeled sample $(\mathbf{x}, y)$ as

$$m_l(\mathbf{x}, y) = p(y|\mathbf{x}) - \max_{\substack{k \in \mathcal{Y} \\ k \neq y}} p(k|\mathbf{x}). \tag{2.14}$$

It is obvious that for a correct classification $m_l(\mathbf{x}, y) > 0$ should hold. Therefore, the generalization error is given by

$$GE = E_{(\mathbf{X},Y)}(m_l(\mathbf{x}, y) < 0), \tag{2.15}$$

where the expectation is measured over the entire distribution of $(\mathbf{x}, y)$. [Breiman, 2001] showed that the generalization error of random forests is upper bounded by

$$GE \leq \bar{\rho} \frac{1 - s^2}{s^2}, \tag{2.16}$$

where $\bar{\rho}$ is the mean correlation between pairs of trees in the forest. The correlation is measured in terms of the similarities of the predictions. s is the strength of the ensemble (*i.e.*, the expected value of the margin over the entire distribution). As can be seen, the size of the correlation of pairs in bagged trees limits the benefits of averaging.

In randomized trees, usually only the node tests are selected randomly; however, their cut-points or thresholds are well tuned. Geurts *et al.* [Geurts et al., 2006] showed that even when selecting the thresholds randomly, thus yielding *extremely randomized trees*, a random forest is able to achieve highly accurate results. Moreover, since the random selection of thresholds further decreases the correlation of the trees Guerts *et al.* suggested to leave out the bagging part and to provide each tree with the entire training set.

Meanwhile, random forests have become increasingly popular and are frequently applied in practice. Their popularity is due to several reasons: RFs have demonstrated to be better or at least comparable to other state-of-the-art methods in both classification [Breiman, 2001] and clustering [Moosmann et al., 2006]. Breiman showed that random forests can be also used for learning distance functions and in order to perform regression tasks. Caruana *et al.* [Caruana et al., 2008] empirically demonstrated that RFs outperform most state-of-the-art learners on high dimensional data problems. Analyzing random forests theoretically in order to explain their power is less straight-forward than, for instance, analyzing boosting or SVMs because they are not based on similar clean mathematical formulations. [Breiman, 2001] tries to give some theoretical justification in terms of the rule of large numbers but no detailed proofs are delivered. More recently, Lin and Jeon [Lin and Jeon, 2006] gave a plausible explanation by showing that random forests can be viewed as an approximated adaptively weighted k-nearest neighbor method. For more detailed discussions and proofs we refer the reader to [Breiman, 2001, Lin and Jeon, 2006].

Especially, the speed in both training and evaluation of randomized trees is one of their main appealing properties. Additionally, RFs can easily be parallelized, which makes them interesting for multi-core and GPU implementations [Sharp, 2008]. RFs are inherently multi-class, therefore it is not necessary to build several binary classifiers for solving multi-class problems. Finally, compared to boosting and other ensemble methods, RFs are more robust against label noise [Breiman, 2001].

These advantages of random forests have also led to increased interest in the computer vision domain. For instance, recently Gall and Lempinsky [Gall and Lempinsky, 2009] presented an efficient object detection framework based on random forests. Shotton *et al.* [Shotton et al., 2008] presented a real-time algorithm for semantic segmentation based on randomized trees. Bosch and Zisserman used RFs for object categorization [Bosch et al., 2007]. Randomized trees have also successfully been applied to visual tracking using keypoints [Lepetit and Fua, 2006].

2.4.2.1 Random Naïve Bayes and Ferns

The success of random forests has led to generalizing the idea of random input variable selection to other classification methods that have similar characteristics as decision trees, *i.e.*, fast and simple computation with low bias and high variance. For example, such classifiers can be naïve Bayes' classifiers where the Bayes' conditional model given in Equation (2.5) is approximated assuming the input feature variables as independent. In more detail, we can write Equation (2.5) in form of

$$p(y|f_1,\ldots,f_N) = \frac{p(y)p(f_1,\ldots,f_N|y)}{p(f_1,\ldots,f_N)}, \tag{2.17}$$

where f_i are the conditionally dependent input feature variables. Since this feature dependency is a problem in practice, *e.g.*, due to a large N, the naïve Bayes' assumption is simply to consider all of the features to be conditionally independent. Hence, we can write

$$p(y|f_1,\ldots,f_n) = p(y)\prod_{m=1}^{M} p(f_i|y), \tag{2.18}$$

where we the denominator of Equation (2.17) is assumed to be constant and can thus be skipped. The idea of random naïve Bayes' [Prinzie and den Poel, 2007] is now to form a committee of bagged naïve Bayes' classifiers, where the features for each classifier comprise a randomly selected subset of the entire feature input space. Such a classifier can finally be written as

$$p(y|\mathbf{x}) = \sum_{t=1}^{T}\prod_{m=1}^{M} p^t(y|f_i^t). \tag{2.19}$$

[Prinzie and den Poel, 2007] showed that this approach is able to match the performance of randomized trees on some data sets; however, being even faster.

Later Ozuysal *et al.* [Özuysal et al., 2007] highlighted that assuming all features to be independent as in naïve Bayes' and totally ignoring the correlations among the features, although fast, can cause low accuracy results, especially in computer vision applications. Therefore, they introduced an approach called *randomized ferns* that is similar to *semi-naïve Bayes* approaches [Zheng and Webb, 2005] and is thus somewhere between naïve Bayes and the original Bayes' theorem. In detail, random ferns consist of M groups of size $S = \frac{N}{M}$, where for each fern the joint probability of its features is calculated; however, the entire ensemble is combined using the independence assumption among the ferns as

$$p(f_1,\ldots,f_n|y) = \prod_{k=1}^{M} p(F_k|y), \tag{2.20}$$

where $F_k = \{f_{k,1}, f_{k,S}\}, k = 1, \ldots, M$ represents the k^{th} fern. The method has shown to perform comparable to random forests and to yield excellent results on the task of visual tracking using keypoints.

2.5 Summary

In this chapter, we have introduced the basic notations and formulations which are used throughout this thesis. We have further reviewed off-line and on-line learning and discussed the differences of the two learning paradigms. As the main focus of this work lies on ensemble methods, we have given a detailed introduction to this family of learning algorithms. In particular, we discussed boosting and random forests as these two methods are the baseline approaches for the rest of the thesis. In the following chapter, we will formally introduce semi-supervised learning and discuss related work within discipline.

Chapter 3

Overview of Semi-Supervised Learning

As we have seen in the previous chapter, there in principle exist two basic paradigms on how to learn from training data. In the first, *supervised learning*, training samples are provided together with their corresponding class labels and in the second, *unsupervised learning*, training samples are given without their labels.

Semi-supervised learning (See also [Chapelle et al., 2006, Zhu and Goldberg, 2009] and for a good overview.) is somewhere between supervised and unsupervised methods, thus learns a classifier from both labeled $\mathcal{D}^L$ and unlabeled $\mathcal{D}^U$ data. Usually, one assumes that there are much more unlabeled samples available than labeled, *i.e.*, $|\mathcal{D}^U| \gg |\mathcal{D}^L|$. Besides classification, semi-supervised learning (SSL) is also used for clustering and regression. In this work, however, we mainly concentrate on classification tasks.

In general, SSL methods can be split up in either *transductive* or *inductive* approaches. Transductive learning aims to train a good classifier for the unlabeled training data in form $H : \mathcal{X}^{l+u} \rightarrow \mathcal{Y}^{l+u}$, which means that the unlabeled data exploited during training are also the test data. Note that this does not necessarily mean that the error for samples that were not observed during training is also low.

Contrary, in inductive learning given both labeled $\{(\mathbf{x}_i, y)\}_{i=1}^{l}$ and unlabeled training data $\{(\mathbf{x}_i)\}_{i=1}^{u}$ the goal is to train a classifier $H : \mathcal{X} \rightarrow \mathcal{Y}$ with low generalization error. However, as we will see later in the literature the terms *inductive* and *transductive* are frequently mixed up and there exist methods, *e.g.*, transductive support vector machines (TSVM) [Vapnik, 1998, Joachims, 1999], that are in their nature inductive learners. Additionally, we will also see that there are ways to convert transductive approaches into inductive ones.

3.1 Relations to Cognitive Science and Biology

It is clear that humans observe the environment mostly without getting any labels for what they see, hear or smell. Occasionally, of course, they do get labels, especially during childhood. Hence, there seems to be a connection on how humans tend to learn and the machine learning perspective of semi-supervised learning. The investigation on how both humans and machines learn may lead to intertwined answers. Additionally, the question arises if an improved understanding of how humans learn from both labeled and unlabeled data may also lead to improved machine learning methods.

Although intuitively, parallelisms of the domains must be given, it is a hard task to conduct proofs. Yet, some recent studies give some first insights to how humans are able to exploit huge amounts of unlabeled data. For instance, co-training, being one of the most frequent used SSL algorithms, has similarities to learning of humans. It has been shown [Bahrick et al., 2002] that the redundancy that co-training exploits is also exploited by humans; *i.e.*, infants exploit the redundancy of their sensors in order to guide the development of selective attention, perception and cognition.

Another recent study [Zaki and Nosofsky, 2007] showed that humans tend to change a model learned from labeled data once they see the unlabeled test data. In particular, Zaki *et al.* showed that humans change their classification boundaries according to unlabeled data, something which is a typical SSL assumption. The same phenomenon was also observed by Zhu *et al.* [Zhu et al., 2007]. For more complicated tasks, however, it was also shown that unlabeled data does not always help humans in order to improve their performance on a given task [Vandist et al., 2009].

Another interesting arising research domain also inspired by the human learning nature is *curriculum learning*; *i.e.*, providing data in a well defined order so that it is easier to assimilate. Curriculum learning builds on the fact that humans get easy data provided first and then successively go forward to more complicated tasks. Recent work [Bengio et al., 2009] has shown that such strategies can also improve learning algorithms and that also semi-supervised learning methods might benefit from such a strategy, which overall suggests that curriculum learning is a promising research direction for future machine learning methods.

3.2 Why does SSL work?

In order to understand why semi-supervised learning works, let us start with a simple example: Consider, a program which has to classify email messages and news, depending on their topic, belonging to either *football* or *ice hockey*. Due to the high overlap in related terms, for instance, “players”, “penalty”, “referee”, *etc.*, it can be very hard for an

unsupervised algorithm to find the right partitions between the two terms. For example, an unsupervised method, *e.g.*, a clustering algorithm, could find many different ways to partition the data. For instance, it can find clusters for "world cup" and "penalty", *etc.*. Contrary, adding a supervised term to the learning will ease findings of discriminative features between the *football* and *ice hockey* class, for instance, "stick", "ball", "Messi", "Gretzky", *etc.*. Clearly, these discriminative features will also allow for better exploiting the unlabeled samples. Hence, an intuitive explanation suggests that by learning from the labeled data, the algorithm gets a better guideline on what it should focus while exploiting the unlabeled data.

In principle, in SSL one wants to improve a classifier $H : \mathcal{X} \rightarrow \mathcal{Y}$ via incorporating large amounts of unlabeled data compared to having only a limited amount of labeled samples. However, since there is no direct link between the distribution $P(\mathbf{x})$ and the target labels $\mathcal{Y}$ most approaches assume an underlying structure that correlates the unlabeled data with some class label and, hence, makes them informative. In order to make classifier learning benefit from additional unlabeled data we thus have to impose certain assumptions about how $P(\mathbf{x})$ influences target labels. For instance, one of the most frequently used assumptions is that the real decision boundary lies in regions of low density and one can thus exploit additional unlabeled data in order to find these regions. We illustrate this so called *large margin* assumption in Figure 3.1 using a toy example.

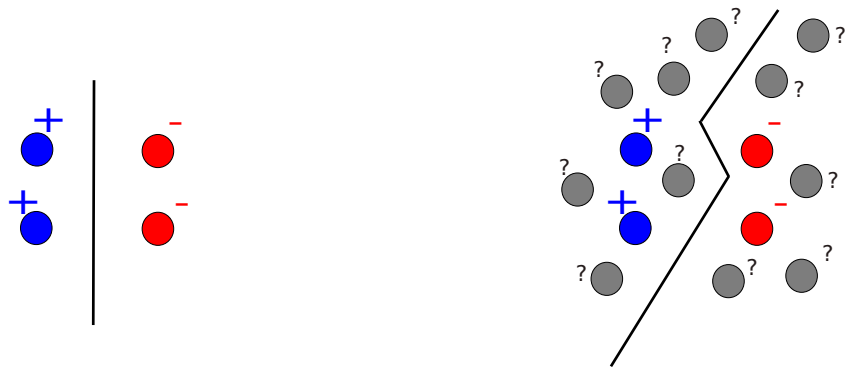


Figure 3.1: Typical SSL assumption: Unlabeled data (grey) helps to change the decision boundary so that it falls into areas of low density (right).

From a loss function perspective, many approaches use these assumptions about $P(\mathbf{x})$ in order to regularize the supervised loss function in the form of:

$$\sum_{(\mathbf{x},y)\in\mathcal{X}_l} \ell(y, h(\mathbf{x})) + \lambda \sum_{\mathbf{x}\in\mathcal{X}_u} \ell_u(h(\mathbf{x})), \tag{3.1}$$

where $h(\cdot)$ is a binary classifier, and $\ell_u(\cdot)$ encodes the regularizer based on the unla-

beled samples. The regularization paradigm can be further subdivided into two main approaches:

Cluster and Manifold Assumption Some algorithms try to infer the *cluster* or *manifold* structure of the feature space with unlabeled samples and use it as an additional cue for the supervised learning process. Note that cluster and manifold approaches are in principle very similar and often differ only by their used metrics. For example in *cluster kernels* [Chapelle et al., 2003], the cluster assumption states that the target function is locally smooth over subsets of the features space delineated by some property of the marginal density. Approaches based on the manifold assumption assume that the target function lies on a low-dimensional manifold. *Label Propagation* [Zhu and Ghahramani, 2002] or *Laplacian SVMs* [Belkin et al., 2006]. The latter two are graph-based methods where ℓ_u has the form of

$$\ell_u(h(\mathbf{x})) = \sum_{\substack{\mathbf{x}' \in \mathcal{X}_u \\ \mathbf{x}' \neq \mathbf{x}}} s(\mathbf{x}, \mathbf{x}') \|h(\mathbf{x}) - h(\mathbf{x}')\|^2, \tag{3.2}$$

where $s(\mathbf{x}, \mathbf{x}')$ is a similarity function. Using this regularization term, one can enforce the classifier to predict similar labels if the samples are similar. While graph-based methods are quite powerful, the pair-wise terms increase their computational complexity.

Another interesting approach, termed expectation regularization (ER), was proposed by Mann *et al.* [Mann and Mccallum, 2007] and developed further for semi-supervised boosting by Saffari *et al.* [Saffari et al., 2008, Saffari et al., 2009a], which improves both computational efficiency and robustness compared to previous methods. ER is a method for exponential-family parametric models where the basic idea is to augment the label-likelihood objective function with a term that encourages the model predictions on unlabeled data to match prior expectations (*e.g.*, label priors).

Large Margin Approaches Another class of methods such as *Transductive Support Vector Machines* (TSVM) [Joachims, 1999, Sindhwani et al., 2006], tries to maximize the margin of the unlabeled samples by avoiding dense regions of the feature space for the decision boundary. For example, variants of Transductive Support Vector Machines maximize the margin for the unlabeled samples by

$$\ell_u(h(\mathbf{x})) = \max(0, 1 - |h(\mathbf{x})|). \tag{3.3}$$

In general, there is no strict separation between the different SSL assumptions and often they overlap and are used simultaneously. For instance, in the literature the cluster and manifold assumption are sometimes treated as separate ideas and sometimes as being

the same. The same holds for the large margin assumption along with the cluster assumption. In later sections we will take a closer look on these assumptions and the derived approaches.

When does SSL not work? As has been discussed above, unlabeled data can only help if there exists a link between the marginal distribution $P(\mathbf{x})$ and the target function to be learned and for learning algorithms that choose the right assumptions on how $P(\mathbf{x})$ is connected with the conditional distribution $P(y|\mathbf{x})$. In fact, if there does not exist such a link or the wrong assumptions are applied, semi-supervised learning can even lead to worse results. There have been several works that theoretically underpinned this issue, *e.g.*, [Balcan and Blum, 2005, Kääriäinen, 2005, Rigollet, 2007, Lafferty and Wasserman, 2008, Ben-David et al., 2008].

Recently, Singh *et al.* [Singh et al., 2009] presented a finite sample analysis method in order to characterize the value of unlabeled data. In more detail, they showed that semi-supervised learning can improve the performance of a supervised learning task if the complexity of the distribution under consideration is too high to be learned using n labeled data points, but is small enough to be learned using $m \gg n$ unlabeled data points. However, it is still an open question if SSL can also improve the accuracy if n is large and, additionally, there is still no guarantee that exploiting unlabeled data helps to improve the classification performance. For more theoretical details about when SSL can fail and when not, we refer the reader to the cited references of this paragraph. In the following, we will look more into detail of some of the most popular SSL approaches.

3.3 Self-Training

Self-training is the most naïve approach to semi-supervised learning. In principle, these methods are *wrapper* methods, *i.e.*, they allow supervised learning methods to be applied to a semi-supervised learning task. In self-training, first a supervised classifier H is learned using only the labeled training data $\mathcal{D}^L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, where l is short for $|L|$, then this classifier is used to predict the labels for the unlabeled data $\mathcal{D}^U = \{(\mathbf{x}_j)\}_{j=l+1}^{l+u}$, where u is short for $|U|$. The classifier is re-trained using $\mathcal{D}^L \cup \mathcal{D}^U$ with $\mathcal{D}^U = \{(\mathbf{x}_j, y_j^*)\}_{j=l+1}^{l+u}$, where y_j^* are self-predicted labels. For the re-training step, either a most confident sub-set or the entire unlabeled data can be used.

Self-training may work well if its underlying assumption, *i.e.*, at least the high confidence predictions are correct, tend to be true. For easy separable data, this is often the case. However, if H makes many errors on predicting the labels, these errors may easily be accumulated over the iterations and lead to failure of the method. Even if only few labels are predicted wrongly, self-training can fail if supervised learners are applied that are

highly susceptible to class-label noise, such as boosting methods (See also section 5.2).

3.4 Generative Methods

In principle, by looking only at unlabeled data, what we can get is an estimate for the marginal data distribution $P(\mathbf{x})$. If we know how the instances from each class are distributed, we can decompose the mixture into individual classes and apply such *mixture models* to semi-supervised learning.

Generative models perform classification by finding good estimates for $p(\mathbf{x}|y)$ and $p(y)$. The class conditional $p(\mathbf{x}|y)$ can be estimated using some model parameters, *e.g.*, the mean μ and covariance matrix σ of a Gaussian distribution. $p(y)$ has to be estimated for K classes. All parameters in $p(\mathbf{x}|y)$ and $p(y)$ can be summarized in one vector θ. Given the training data $\mathcal{D}$, during training generative models try to find good estimates for θ using the maximum likelihood estimate (MLE)

$$\hat{\theta} = arg\max_{\theta} p(\mathcal{D}|\theta) = arg\max_{\theta} \log\ p(\mathcal{D}|\theta) \tag{3.4}$$

Note that the log likelyhood $\log\ p(\mathcal{D}|\theta)$ is often preferred to estimating the likelihood directly because it is easier to handle. When we rewrite the log likelihood as follows

$$log\, p(\mathcal{D}|\theta) = \log \prod_{i=1}^{l} p(\mathbf{x}_i, y_i|\theta) = \sum_{i=1}^{l} p(y_i|\theta)p(\mathbf{x}_i|y_i, \theta), \tag{3.5}$$

the MLE can be easily found using constrained optimization.

For the semi-supervised learning problem, where $\mathcal{D} = \mathcal{D}^L \cup \mathcal{D}^U$, the log likelihood function changes to

$$\begin{aligned} \log\ p(\mathcal{D}|\theta) =& log\Big(\prod_{i=1}^{l} p(\mathbf{x}_i, y_i|\theta) \prod_{i=l+1}^{l+u} p(\mathbf{x}_i|\theta)\Big) \\ =& \sum_{i=1}^{l} \log p(y_i|\theta)p(\mathbf{x}_i|y_i, \theta) + \sum_{i=l+1}^{l+u} \log p(\mathbf{x}_i|\theta). \end{aligned} \tag{3.6}$$

The task of a semi-supervised algorithm is now to find the MLE of Equation 3.6 which needs to fit both the labeled and the unlabeled instances. Note that since the labels for the unlabeled samples are not given, they become additional optimization variables, which makes the overall optimization problem non-convex and thus difficult.

Expectation Maximization In generative SSL methods, one of the most frequent optimization methods used is the *expectation maximization* (EM) algorithm. If the training data given is $\mathcal{D} = \mathcal{D}^L \cup \mathcal{D}^U$, then the missing (hidden) variables are $\mathcal{H} = \{y_{l+1}, \ldots, y_{l+u}\}$. The EM algorithm is an iterative method to find the model parameters θ that locally maximize $p((D)|\theta)$. EM in each iteration consists of two steps, an expectation step (E-step) and a maximization step (M-step). It keeps a distribution $q_t(\mathcal{H})$ over the hidden variables. In practice, EM was used for many SSL problems, *e.g.*, text classification [Nigam et al., 2006], *etc.*. However, since it is a local optimizer, it can get stuck in local minima. We depict the EM method in detail in Algorithm 3.1.

Algorithm 3.1 Expectation Maximization

Require: Labeled data $\mathcal{X}_l$ and unlabeled data $\mathcal{X}_u$
Require: Initial parameter θ_0
repeat
E-step: compute $q_t(\mathcal{H}) = p(\mathcal{D}|\theta, \theta_t)$
M-step: find θ_{t+1} that maximizes $\sum_{\mathcal{H}} q_t(\mathcal{H}) \log\, p(\mathcal{D}, \mathcal{H}|\theta_{t+1})$
$t = t + 1$
until $p(\mathcal{D}|\theta_t)$ converges
Output the final parameters θ.

3.5 Co-Training and Multi-View Learning

Co-training [1] [Blum and Mitchell, 1998] which exploits the redundancy of unlabeled input data is another popular SSL method. In co-training, two initial classifiers h_1, h_2 are trained on some labeled data $\mathcal{D}^L$ using different redundant "views". Different views can be, for instance, different types of uncorrelated features. Then, one classifier updates the other one on samples of the unlabeled data set $\mathcal{D}^U$ where it is most confident. Co-training is a wrapper method, which means it does not matter which learning algorithms are applied as long as they are able to deliver confidence-rated predictions. We depict the algorithmic steps in Algorithm 3.2.

The approach has shown to converge if two conditions hold:

1. There exist two separate views $\mathbf{x} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}]$ and the task is solvable under each view.

2. The views should be conditionally independent given the class label; *i.e.*, $P(\mathbf{x}^{(1)}|y, \mathbf{x}^{(2)}) = P(\mathbf{x}^{(1)}|y)$ and $P(\mathbf{x}^{(2)}|y, \mathbf{x}^{(1)}) = P(\mathbf{x}^{(2)}|y)$.

[1] A.K.A. collaborative bootstrapping or multi-view learning.

Algorithm 3.2 Co-Training [Blum and Mitchell, 1998]

Require: Labeled data $\mathcal{X}_l$ and unlabeled data $\mathcal{X}_u$
Require: Two learners $F_1(\mathbf{x})$ and $F_2(\mathbf{x})$
Require: Set max iterations T
for $t = 1, \ldots, T$ **do**
Train $F_1(\mathbf{x})$ on $\mathcal{X}_l$ using only a subset $\mathbf{x}_1$ of $\mathbf{x}$
Train $F_2(\mathbf{x})$ on $\mathcal{X}_l$ using only a subset $\mathbf{x}_2$ of $\mathbf{x}$
Let $F_1(\mathbf{x})$ label p positive and n negative samples from $\mathcal{X}_u$
Let $F_2(\mathbf{x})$ label p positive and n negative samples from $\mathcal{X}_u$
Add the self-labeled samples to $\mathcal{X}_l$
end for
Output the final classifier as $F(\mathbf{x}) = F_1(\mathbf{x}) + F_2(\mathbf{x})$.

Co-training iteratively makes the two (diverse) classifiers agree on the predictions on the unlabeled data. Note that this increases the decision margin and is hence another explanation why the unlabeled data can help improving the classification accuracy in such scenarios. [Dasgupta et al., 2002] showed that when the two views are sufficient and conditionally independent, the generalization error of co-training is upper-bounded by the disagreement between the two classifiers. Zhou *et al.* [Zhou and Xu, 2007] showed that in principle a single labeled training sample can be sufficient for co-training to converge successfully.

One of the main limitations of co-training is the condition that the two sets have to be conditionally independent given the class in order to converge. This condition was later relaxed by [Balcan et al., 2004], but only by introducing another condition which is "the classifiers must never be confident but wrong". Meanwhile, there exist improvements of the original algorithm in terms of robustness, *e.g.*, [Leskes and Torenvliet, 2008, Shen et al., 2005] and [Wang and Zhou, 2007] provided a PAC-style proof that co-training can even work if conditional independence is not given.

Another problem in co-training arises when the observations of the views can be noisy. For instance, consider one wants to build a co-training system which should classifier vehicles in either cars or trucks, respectively. One solution for such a task would be to train a classifier on audio signals and another one based on visual appearance. However, in case of occlusions – *e.g.*, a truck occludes a car – the visual classifier would correctly yield "no car detected" but the audio classifier may detect that acoustic signal of the car's engine. In such a case, both classifiers are in principle making the correct predictions but disagree on the assigned labels. If this happens frequently during the learning process, typical co-training would fail. However, recently, Christoudias *et al.* [Christoudias et al., 2008a, Christoudias et al., 2008b] showed that this problem of noise and disagreement in

multi-view learning can be tackled via incorporating conditional entropy measures into the learning and by applying Gaussian processes, respectively.

When we talk about co-training we mainly refer to the case when two classifiers train each other. If more than two classifiers are used, we denote the learning approach as *multi-view learning*. For example, tri-training [Zhou and Li, 2005] is an extension of co-training that uses three classifiers instead of two. [Zhou and Li, 2005] has shown both theoretically and empirically that tri-training has weaker conditions in order to converge, which makes it more applicable in practice. However, [Blum and Mitchell, 1998] argued in their co-training approach that there are not many candidate classifiers that can agree on unlabeled data in two views. In principle this is positive because a small hypothesis space, which also fits the labeled data well, is less likely to overfit. While taking more classifiers decreases the independence conditions among the classifiers, it is simultaneously harder to find multiple views where each of them is strong enough in order to deliver reliable votes on the unlabeled data.

3.6 Graph-based Methods

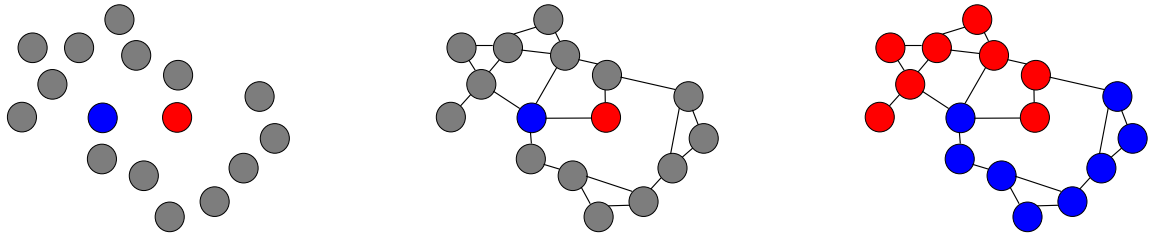
Graph-based semi-supervised learning methods assume the entire training set $\mathcal{D}$ consisting of both labeled $\mathcal{D}^{\mathcal{L}}$ and unlabeled samples $\mathcal{D}^{\mathcal{U}}$ as fully connected graph $g = \{(V, E)\}$. According to this principle, labeled and unlabeled samples are represented as nodes and edges represent similarities or distances among the samples, respectively. Given such a graph, the main assumption is that the labels are smooth with respect to the graph, such that they vary slowly on the graph. If two instances are connected by a strong edge, *i.e.*, they are considered to being highly similar, their labels should be the same. The similarities can be encoded in a weight matrix $\mathbf{W}$, where $\mathbf{W}_{ij}$ is non-zero if $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors. A popular weight matrix is the Gaussian kernel or radial basis function (RBF)

$$\mathbf{W}_{ij} = \exp\Big(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\Big), \tag{3.7}$$

where σ is the kernel bandwidth. Known labels are used to propagate information through the graph in order to label all the nodes.

Based on the graph representation of the data, there exist several different ways to perform semi-supervised learning:

Graph-based SSL with harmonic functions Zhu *et al.* [Zhu et al., 2003] presented a graph-based SSL framework based on Gaussian random fields and harmonic functions. A harmonic function is a function that has the same values as given labels on the labeled data and satisfies the weighted average property on the unlabeled data. This means that an



(a) (b) (c)

Figure 3.2: Graph-based semi-supervised learning: Given some labeled and unlabeled data (a), we can create a connected graph (b). Given this graph, the final solution looks as illustrated in (c).

unlabeled node gets assigned with the labels of its neighbors' values. In particular, Zhu *et al.* compute a real-valued function $f : V \to \mathbb{R}$ on a given graph $G = (V, E)$ and assign labels to the unlabeled data using f. They form a Gaussian random field $p_\beta(f) = \frac{e^{-\beta E(f)}}{Z_\beta}$ over the quadratic energy function

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij}(f(i) - f(j))^2 \tag{3.8}$$

and the partition function $Z_\beta = \int_{f|L=f_l} \exp(-\beta E(f)))df$. The function that minimizes Equation (3.8) satisfies $\Delta f = 0$ and is thus called *harmonic*. The harmonic characteristic enforces that the value of f for an unlabeled sample is the average of its neighboring points. Δ is the *combinatorial Laplacian*, in particular, given as $\Delta = D - W$ where $D = diag(d_i)$ with entries $d_i = \sum_j w_{ij}$ and $W = [w_{ij}]$ is the weight matrix.

[Zhu et al., 2003] showed that there exist a closed-form and globally optimal solution for the *harmonic energy minimization* and highlighted the tight connections to electrical networks and random walks. Similar to the work of Zhu *et al.* there exist various semi-supervised approaches exploiting graphs, for example, using low density separation [Chapelle and Zien, 2005] or approaches based on mincut [Blum and Chawla, 2000] and randomized mincut [Blum et al., 2001].

Manifold Assumption Graph-based SSL based on harmonic functions has two drawbacks: first, it is a transductive method and second the labels of labeled instances are fixed, which can cause troubles in case of class label noise. Semi-supervised manifold regularization [Belkin et al., 2006] addresses these issues. In general, a manifold is a topological space that is locally Euclidean. The points in a small neighborhood are expressed as coordinates in Euclidean spaces using so called coordinate charts or patches.

The differences of the coordinate charts can be expressed using transition maps or transition functions [Weisstein, 2002].

Definition 3.1: *A d-dimensional manifold* $\mathcal{M} = \bigcup_{\alpha} \mathsf{U}_{\alpha}$ *is a mathematical object that generalizes domains in* $\mathbb{R}^d$*. Each one of the "patches"* U_{α} *that cover* $\mathcal{M}$ *is endowed with a system of coordinates* $\alpha : \mathsf{U}_{\alpha} \mapsto \mathbb{R}^d$*.*

If two patches U_{α} and U_{β} overlap, the transition functions $\beta \circ \alpha^{-1} : \alpha(\mathsf{U}_{\alpha} \bigcap \mathsf{U}_{\beta}) \mapsto \mathbb{R}^d$ must be smooth, *i.e.*, infinitely differentiable. A *Riemannian Manifold* inherits from its local system of coordinates most geometrical notions available on $\mathbb{R}^d$, such as metrics, angles, volumes, *etc.*.

In manifold regularization, f is defined over the entire feature space $f : \mathcal{X} \mapsto \mathbb{R}$ and is regularized with respect to the graph Laplacian Δ. In more detail, a manifold regularizer can be written as

$$\Omega(f) = \lambda_1 ||f||^2 + \lambda_2 \mathbf{f}^T L \mathbf{f}, \tag{3.9}$$

where $||f||^2 = \int_{x \in \mathcal{X}} f(x)^2 dx$ is an additional regularization term which enforces smoothness in order to improve generalization performance and $\lambda_1, \lambda_2 \geq 0$ are the convex weightings of the two regularization terms. Given an arbitrary loss function, for instance, $\ell(\mathbf{x}, y, f(x)) = (y - f(x))^2$, the complete problem can be written as

$$\min_{f : \mathcal{X} \mapsto \mathbb{R}} \sum_{i=1}^{l} (y_i - f(x))^2 + \lambda_1 ||f||^2 + \lambda_2 \mathbf{f}^T L \mathbf{f}. \tag{3.10}$$

Having the problem rewritten in terms of manifold regularization, various learning methods have been applied to learn classifiers from both labeled and unlabeled data, for instance, boosting, support vector machines and least-squares. Although there have been many successful SSL algorithms based on the manifold assumption, they do not work if the given data lies on a mixture of manifold, which occurs frequently in practice where manifolds tend to intersect and partially overlap. Recently, Goldberg *et al.* [Goldberg et al., 2009] studied the problem of semi-supervised learning in multi-manifold scenarios. Their algorithm works by combining the single manifold and the cluster assumption in order to find piecewise smooth parts of the target function.

Xu *et al.* [Xu et al., 2009] presented a framework for features selection via manifold regularization. In their method, an optimal subset of features is identified by maximizing a performance measure that combines the classification margin with manifold regularization. In particular, the feature selection is formulated as a convex-concave optimization problem, where the saddle point holds the optimum.

3.7 Boosting and SSL

There exist several approaches to semi-supervised learning with boosting, *e.g.*, [Collins and Singer, 1999, Bennett et al., 2002, d'Alche Buc et al., 2002]. The main idea in most of these approaches is to add an unsupervised regularization term to the supervised loss function of boosting that penalizes decision boundaries passing through high density regions. Similar to Equation (3.1) semi-supervised boosting loss functions look as follows

$$\begin{aligned}\mathcal{L}(\mathbf{X}) &= \mathcal{L}_l(\mathcal{X}_l) + \beta\mathcal{L}_u(\mathcal{X}_u) \\ &= \sum_{(\mathbf{x},y)\in\mathcal{X}_l} \ell_l\left(yH(\mathbf{x})\right) + \beta \sum_{\mathbf{x}\in\mathcal{X}_u} \ell_u(H(\mathbf{x})),\end{aligned} \tag{3.11}$$

where $\mathcal{L}_l$ and $\mathcal{L}_u$ are the loss functions for the labeled and unlabeled samples, respectively. Note that this is similar to TSVMs and related algorithms that exploit the large margin assumption. In the following, we will shortly review some more recently proposed semi-supervised boosting methods.

3.7.1 SERBoost

Mann and McCallum [Mann and Mccallum, 2007] analyzed many SSL algorithms and pointed out that although there exist a vast amount of SSL approaches, most of them need tedious parameter tuning and have bad scaling behavior of $\mathcal{O}(n^3)$, where n is the number of unlabeled samples, which makes them hardly applicable in practice. In order to tackle these issues, they proposed the *Expectation Regularization* (XR) method which augments a log-likelihood objective function with a term that penalizes model predictions on unlabeled data that deviate from certain expectations. Algorithms based on XR scale very well and do not need complicated parameter tuning.

Based on the idea of expectation regularization, Saffari *et al.* [Saffari et al., 2008] proposed a semi-supervised boosting algorithm called *SERBoost*. SERBoost assumes a given prior conditional probability in form $P_P(y|\mathbf{x})$ with $y \in \{-1, +1\}$) and $\mathbf{x} \in \mathbb{R}^d$. The prior distribution is used in order to impose certain expectations over the labels of the unlabeled samples. Similar to previous SSL boosting approaches, the loss function is written as

$$\mathcal{L}(H(\mathbf{x}), \mathcal{X}) = \mathcal{L}_l(H(\mathbf{x}), \mathcal{X}_l) + \beta\mathcal{L}_u(H(\mathbf{x}), \mathcal{X}_u), \tag{3.12}$$

where $\mathcal{X} = \mathcal{X}_\mathcal{L} \cup \mathcal{X}_\mathcal{U}$, $\mathcal{X}_\mathcal{L} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ and $\mathcal{X}_\mathcal{L} = \{(\mathbf{x}_1), \ldots, (\mathbf{x}_u)\}$ and $\beta \geq 0$ defines the importance of the unlabeled loss. For the labeled loss the exponential formulation is used to be $\mathcal{L}_l(H(\mathbf{x}), \mathcal{X}_l) = \sum_{\mathbf{x}\in\mathcal{X}_L} e^{-yH(\mathbf{x})}$. For the unlabeled loss the

Kullback-Leibler (KL) divergence is used between the prior probability and learned distribution to be $\mathcal{L}_u(H(\mathbf{x}), \mathcal{X}_u) = \mathbb{E}(D(P_P||\hat{P}))$. For unlabeled samples pseudo labels are introduced to be $y_p = 2P_p^+(\mathbf{x}) - 1$. Finally, the overall loss function can be written as

$$\mathcal{L}(H(\mathbf{x}), \mathcal{X}) = \sum_{\mathbf{x}\in\mathcal{X}_L} e^{-yH(\mathbf{x})} + \sum_{\mathbf{x}\in\mathcal{X}_U} e^{-y_pH(\mathbf{x})} cosh(H(\mathbf{x})). \tag{3.13}$$

For learning the boosting model Saffari *et al.* use a boosting approach that performs gradient descent in function space [Mason et al., 1999]. The method as shown experimentally to deliver good results using various sources of priors.

3.7.2 Boosting with Manifold Regularization

Another way to perform semi-supervised boosting is to exploit the manifold assumption as used in *ManifoldBoost* proposed by [Loeff et al., 2008]. ManifoldBoost regularizes the loss function by incorporating unlabeled data using the graph Laplacian L and learns the model using GradientBoost. In detail, the loss function used in ManifoldBoost is written as

$$\mathcal{L}(H(\mathbf{x}), \mathcal{X}) = \sum_{\mathbf{x}\in\mathcal{X}_L} \ell(y_i, H(\mathbf{x}_i)) + \sum_{i,j} H(\mathbf{x}_i)L_{i,j}H(\mathbf{x}_j). \tag{3.14}$$

The unlabeled loss enforces labels of an unlabeled sample to be the average of its K nearest neighbors and yields a final classifier ensemble which is smooth on the underlying manifold. Note that there exist also other SSL boosting approaches working on the manifold assumptions, such as [Kegel and Wang, 2005, Chen and Wang, 2008].

3.7.3 Co-training and Boosting

There also exist approaches that use boosting in order to perform co-training. While [Collins and Singer, 1999] use co-training in the original setting as proposed in [Blum and Mitchell, 1998], *i.e.*, co-training is the meta algorithm for two common boosters, Leskes and Torenvliet [Leskes and Torenvliet, 2008] proposed a new boosting algorithm that is more directly targeted towards the co-training principle. In particular, the method, AgreementBoost, is based on the observation that co-training works by enforcing two diverse classifiers to agree on their predictions on the unlabeled samples which increases the generalization margin of the combined classifier. In order to enforce agreement during the boosting iterations Leskes and Torenvliet added a regularization term to the boosting loss that penalizes disagreement of the weak learners on the unlabeled samples:

$$\mathcal{L}(H(\mathbf{x}), \mathcal{X}) = \sum_{\mathbf{x}\in\mathcal{X}_L} \ell(y_i, H(\mathbf{x}_i)) + \sum_{\mathbf{x}\in\mathcal{X}_U} V(\mathbf{x}_j), \tag{3.15}$$

where $V(\mathbf{x}_j)$ is a term penalizing high variance over the unlabeled data.

3.8 Computer Vision Applications

In computer vision, the probably most frequently applied semi-supervised learning algorithm is co-training. For example, Levin *et al.* [Levin et al., 2003] used co-training to train a car detector. They start with a small number of hand labeled samples and generate additional labeled examples by applying co-training of two boosted off-line classifiers, where one uses gray-value images and the other is trained from background subtracted images, respectively. Moreover, Javed *et al.* [Javed et al., 2005] applied an arbitrary number of classifiers and extended the method to on-line learning. In particular, they first generate a seed model by off-line boosting, which is improved later on by on-line boosting. If multiple disjoint views exist, co-training can also be applied for tracking, *e.g.*, [Tang et al., 2007, Yu et al., 2008, Liu et al., 2009]. There also exist several approaches based on deep neural networks in order to improve the visual recognition performance using unlabeled data, *e.g.*, [Yu et al., 2008, Mobahi and Collobert, 2009]. Recently, Fergus *et al.* [Fergus et al., 2009] presented a semi-supervised framework that is able to learn object classifiers from 80 million images. In particular, they propose a graph-based method that scales linear with the number of samples and thus allows for large-scale usage. Guillaumin *et al.* [Guillaumin and Schmid, 2010] proposed a multimodal SSL approach used for image categorization, where the main idea is to additionally to the visual information, also exploit other sources of information such as text, which is surrounding images on web pages. Socher and Fei-Fei [Socher and Fei-Fei, 2010] applied Semi-supervised learning to image-segmentation.

3.9 SSL from weakly related data

As has been shown above, there exist a large amount of methods and algorithms for the semi-supervised learning problem. The main differences between these approaches are often only based on their assumptions which they are imposing over the unlabeled data (*e.g.*, manifold assumption or large margin assumption, *etc.*) and on which supervised learning method they are based, such as SVMs or boosting. Yet, one assumption that most of them have in common is that the underlying marginal data distribution $P(\mathcal{X}, \mathcal{Y})$ is *i.i.d.*, which means they draw samples from data of which they assume it is identical and independently distributed. However, in practice, unlabeled data does not necessarily come from the same distribution as the labeled data.

Another problem that occurs in practice but is ignored by most approaches is the fact that although unlabeled data is usually easy to obtain, unlabeled data which consists of sufficient amounts of target class samples is not. For instance, consider the problem of training a visual object detector for alpacas (see Figure 3.3a). For this task, it is difficult

to obtain many labeled images with alpacas. However, it is also difficult to obtain many unlabeled images containing alpacas. The same is true for mongoose (Figure 3.3b) and many other target objects.

a) b)

Figure 3.3: Examples for difficult to obtain images: alpaca (a) and mongoose (b)

A practically useful semi-supervised learning algorithm has to be able to handle weakly related unlabeled data. Unfortunately, there is only a limited amount of approaches that try to tackle this problem and it is also not part of this thesis. However, recently, there have been proposed some first algorithmic attempts that are worth to be mentioned:

Self-taught Learning Raina *et al.* [Raina et al., 2007] highlighted the problem that unlabeled data often does not consist of sufficient samples from the target class and presented a framework called "self-taught learning" or STL. In STL, the main idea is to perform transfer learning from unlabeled data; *i.e.*, although the unlabeled data is not necessarily related to the target class and labels are not available, they show that learning can still benefit from such samples. In particular, STL assumes of having labeled data $\mathcal{D}^L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ drawn *i.i.d.* from some distribution. Additionally, they suppose a set of unlabeled data $\mathcal{D}^U = \{(\mathbf{x}_{l+u}), \ldots, (\mathbf{x}_{l+u})\}$, however, not necessarily drawn from the same distribution as $\mathcal{D}^L$. Yet, although they assume $\mathcal{D}^U$ only weakly related to $\mathcal{D}^L$, they do not assume the unlabeled data to be totally unrelated. The STL consists of two steps: First, an unsupervised learning method, *i.e.*, sparse coding [Olshausen and Field, 1996], is applied to $\mathcal{D}^U$ in order to obtain a higher-level sparse representation of the unlabeled images. This representation is then used in order to train a common SVM on $\mathcal{D}^L$. Although the approach of Raina *et al.* is trivial and seems to be straight-forward, an evaluation of STL on several domains, *e.g.*, object categorization, character recognition and

text classification, showed that even with this low-sophisticated method weakly-related unlabeled data can help improving the classification accuracy.

Semi-Supervised Learning from Weakly-Related Unlabeled Data Building on the similar ideas as self-taught learning, recently Yang *et al.* [Yang et al., 2008] presented an improved version of STL called "Semi-Supervised Learning with Weakly-Related Unlabeled Data" (SSLW). In particular, Yang *et al.* highlight that many SSL approaches are based on the cluster assumption, which, however is violated if the unlabeled data is only weakly related to the target classes. SSLW also tries to find a better data representation that is both informative to the target class and consistent with the feature coherence patterns of the weakly related unlabeled data.

In more detail, out of the labeled data $\mathcal{D}^L$, SSLW uses a document-word matrix $M_D = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_l)$, where $\mathbf{d}_i \in \mathbb{N}^V$ represents the word-frequency vector for document d_i and V is the size of the vocabulary. Additionally, they make use of a second matrix, the word-document matrix G out of both labeled and unlabeled data. $G = (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_V)$, where $g_i = (g_{i,1}, g_{i,2}, \ldots, g_{i,n})$ represents the occurance of the ith word in all the n documents.

For a SVM-formulation one could now use M_D in order to build the kernel $K = M_D^\mathsf{T} M_D$ for the SVM's dual formulation. However, such a kernel would discard weakly related documents, *i.e.*, set the similarity to zero. Therefore, Yang *et al.* augment the kernel with a word-correlation matrix $R \in \mathbb{R}^{V \times V}$ to be $K = M_D^\mathsf{T} R M_D$. In R, R_{ij} represents the correlation between ith and the jth words. The goal is now to find the optimal R that maximizes the categorization margin. This is done by regularizing R according to G by introducing an internal representation of words $W = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_V)$, $\mathbf{w}_i$ is the internal representation of the ith word. The word-correlation matrix can then be written as $R = W^\mathsf{T} W$. Now, the dual formulation of the SVM can be changed to a min-max problem in order to find both the maximum α and minimum R.

$$\min_{R \in \Delta, U, W} \max_{\alpha} \alpha^\mathsf{T} e - \frac{1}{2}(\alpha \circ y)^\mathsf{T} (M_D^\mathsf{T} R M_D)(\alpha \circ y) \tag{3.16}$$

Equation 3.16 can be efficiently solved using Second Order Cone Programming (SOCP) [Boyd and Vandenberghe, 2004]. For text categorization, SSLW has successfully demonstrated of being able to leverage the usage of both labeled and weakly-related unlabeled data in order to increase the generalization error and significantly outperformed self-taught learning and state-of-the-art SSL methods such as TSVM [Bennett and Demiriz, 1999] and manifold regularization [Belkin et al., 2006].

EigenTransfer Self-taught learning and SSLW can both also be considered as a transfer learning problem; however, without knowing the class labels of the source data.

Recently, Dai *et al.* [Dai et al., 2009] proposed a general transfer learning framework, EigenTransfer, were transfer learning from unlabeled data is considered a sub-problem of general transfer learning. In EigenTransfer, the main idea is to create a weighted graph $G = (E, V)$ of both source and target data, where the nodes $V = \{v_i\}_{i=1}^{n}$ represent instances, features and labels and the edges $E = \{e_{ij}\}_{i,j=1}^{n}$ represent the relations between end nodes, connecting the target and the source data. In particular, the weights ϕ_{ij} of the edges are based on the number of co-occurances between the end nodes in the target and the source data. By learning the spectra, *i.e.*, a set of eigenvalues, of this graph, Dai *et al.* obtain an eigen feature representation for all the nodes in the task graph. The new feature representation can then be used in order to transfer knowledge from the source to the target domain.

In case of self-taught learning, in the task graph, the nodes that usually carry the labels of the source data are empty. [Dai et al., 2009] use normalized cut in order to learn the spectra of the graph. In more detail, they calculate the graph Laplacian $L = D - W$ out of the adjacency matrix W and the diagonal matrix D. Then the first N eigenvectors of L build a new feature space where any supervised learning algorithm can be applied. In the experiments EigenTransfer has proven to be a meaningful approach to self-taught learning.

Although the three methods shorty described in this section tried to tackle the problem of semi-supervised learning from unlabeled data that is either weakly related to the target class or the amount of good positive unlabeled data is limited, the problem is yet in its beginnings and still there remains several open questions and there is large room for further investigation.

3.10 Summary

In this chapter, we have introduced the concept of semi-supervised learning and reviewed the current state-of-the-art. We have seen that there exist a vast amount of SSL algorithms, most of them extensions of popular supervised learning methods, and that they are already frequently used in many applications. Note that we left out a detailed review of semi-supervised kernel methods, because this would go beyond the scope of this work. However, we refer the interested reader to [Huang et al., 2006]. Most importantly, we showed that semi-supervised learning works by imposing certain assumptions, *e.g.*, manifold or cluster assumptions, over the unlabeled data and that using the correct assumptions often decides over the success of an algorithm depending on the actual data set. In the following chapters, we will concentrate our further discussion on the usage of ensemble-based learning methods since they have demonstrated to be very powerful and are often used in various computer vision applications.

Chapter 4

SemiBoost and Visual Similarity Learning

As we have motivated in the previous chapter, semi-supervised learning is an increasingly important learning paradigm with various potential practical applications. Additionally, we have seen that there already exist a large amount of different SSL approaches. A large subset of SSL methods, especially those based on graphs and the manifold assumption, rely on a priori given similarities or distance measures which are needed in order to compare samples (labeled and unlabeled) in feature space. It is clear that the accuracy of these similarities determines the success of the semi-supervised learners. One way to obtain good similarities is to learn similarity or distance functions that can then be used for further reasoning or processing.

In the following, we present an approach that combines visual similarity learning and semi-supervised boosting using manifold regularization. In particular, we use a limited amount of labeled samples in order to, first, train a discriminant distance function and, second, use this distance function as a metric in order to guide a variant of SemiBoost [Mallapragada et al., 2009] through exploiting a huge set of unlabeled data.

4.1 Learning Distance Functions

In machine learning problems, the distance metrics are often given in huge matrices that encode standard distances such as the Euclidean or the Chi-Squared. Although this works for many applications it has several fundamental problems: First, the data stored in matrices grows with $\mathcal{O}(n^2)$, where n is the number of samples. Second, it is often hard to decide which is the best similarity metric in order to solve a certain task. Especially in computer vision, it is difficult to determine what makes some digital images similar and

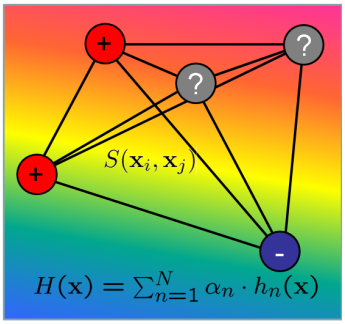


Figure 4.1: SemiBoost uses graph-based regularization in order to train a boosted classifier from both labeled and unlabeled data.

some dissimilar. Finally, if we deal with high-dimensional features spaces, we would like to prefer distance measures that take only a discriminant subset of the features into account which leads to both higher efficiency and accuracy.

In recent years, these arguments have led to approaches that learn distance or similarity functions based on labeled training data, which has the advantage that the discriminant information can be extracted from the eventually high dimensional data that best supports the task and eases further discriminative processing. Hence, learning distance or similarity functions has become an important strand of research within machine learning. In computer vision, there exist several approaches that investigated learning of similarity functions for object categorization in order to handle high intraclass variance and low interclass variance, *e.g.*, [Nowak and Jurie, 2007, Frome et al., 2007, Babenko et al., 2009a, Jain et al., 2008, Jain et al., 2008]. A learned metric can also be plugged into other classifiers, for instance, as a kernel to a SVM or into a knn-classifier [Shakhnarovich et al., 2005]. Metric learning has also been used for image retrieval [Hertz et al., 2004, Torralba et al., 2008] and even for human pose estimation and tracking [Shakhnarovich, 2005].

4.2 SemiBoost

We assume a typical semi-supervised learning setting in form of a labeled data set $\mathcal{X}^L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{|\mathcal{X}^L|}, y_{|\mathcal{X}^L|})\} \subseteq \mathcal{X} \times \mathcal{Y}$ where $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^d$ and $y_i \in \mathcal{Y}$ and an unlabeled data set in form $\mathcal{X}^U = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{X}^U|}\} \subseteq \mathcal{X}$. Additionally, we focus on the binary classification problem, therefore $\mathcal{Y} = \{+1, -1\}$ and the samples are split into two sets $\mathcal{X}^L = \mathcal{X}^+ \cup \mathcal{X}^-$ of all samples with a positive class and the set of all samples with negative class, respectively. The goal is to learn a boosted classifier $H : \mathcal{X} \rightarrow \mathcal{Y}$ which is trained using both labeled and unlabeled samples.

SemiBoost [Mallapragada et al., 2009] is a manifold-based approach and uses the

following unlabeled loss function:

$$\ell_u = \sum_{\mathbf{x}\in\mathcal{X}^U}\Big(C\sum_{\mathbf{x}'\in\mathcal{X}^U} s(\mathbf{x},\mathbf{x}')e^{F(\mathbf{x})-F(\mathbf{x}')} + \sum_{(\mathbf{x}',y')\in\mathcal{X}^L} s(\mathbf{x},\mathbf{x}')e^{-2y'F(\mathbf{x})}\Big), \tag{4.1}$$

where C is a trade-off parameter and $s(\mathbf{x},\mathbf{x}')$ is a similarity measure between two data samples. The unlabeled loss function is a combination of two terms: the first term regularizes only over the unlabeled samples, while the second term uses both labeled and unlabeled data samples. If one uses a symmetric similarity measure, *i.e.*, $s(\mathbf{x},\mathbf{x}') = s(\mathbf{x}',\mathbf{x})$, the unlabeled loss can be simplified as

$$\ell_u = \sum_{\mathbf{x}\in\mathcal{X}^U}\Big(C\sum_{\mathbf{x}'\in\mathcal{X}^U} s(\mathbf{x},\mathbf{x}')\cosh(F(\mathbf{x})-F(\mathbf{x}')) + \sum_{(\mathbf{x}',y')\in\mathcal{X}^L} s(\mathbf{x},\mathbf{x}')e^{-2y'F(\mathbf{x})}\Big). \tag{4.2}$$

As in general manifold-based SSL approaches (Equation (3.2)), $s(\mathbf{x},\mathbf{x}')$ enforces the graph-based smoothness over the data, *i.e.*, that if $\mathbf{x}$ and $\mathbf{x}'$ are very similar also the labels should be the same.

As can be seen, [Mallapragada et al., 2009] ignores the standard loss over labeled data. However, we observed that this term can be informative and, therefore, we propose the following loss function

$$\sum_{(\mathbf{x},y)\in\mathcal{X}^L} e^{-yF(\mathbf{x})} + \sum_{\mathbf{x}\in\mathcal{X}^U}\Big(\lambda_u\sum_{\mathbf{x}'\in\mathcal{X}^U} s(\mathbf{x},\mathbf{x}')\cosh(F(\mathbf{x})-F(\mathbf{x}')) + \lambda_l\sum_{(\mathbf{x}',y')\in\mathcal{X}^L} s(\mathbf{x},\mathbf{x}')e^{-2y'F(\mathbf{x})}\Big), \tag{4.3}$$

where λ_u and λ_l determine how much the unlabeled loss terms can influence the training. In the following, we will use this formulation of SemiBoost.

4.2.1 Learning

The learning of a boosting model consists of finding weak learners $f_i(\cdot)$ and their weights α_i sequentially. This means we need to solve

$$(\alpha_i, f_i) = \arg\min_{\alpha,f} \mathcal{L}, \tag{4.4}$$

where $\mathcal{L}$ is the loss function represented in Equation (4.3).

We now have to solve the following optimization problem

$$\begin{aligned}\arg\min_{f(x),\alpha} = \sum_{\mathbf{x}'\in\mathcal{X}^U}\Big(&\sum_{(\mathbf{x},y)\in\mathcal{X}^L} s(\mathbf{x},\mathbf{x}')e^{-2y(F(\mathbf{x}')+\alpha f(\mathbf{x}'))} \\ &+\lambda_u\sum_{\mathbf{x}'\in\mathcal{X}^U} s(\mathbf{x},\mathbf{x}')e^{((F(\mathbf{x}')-F(\mathbf{x}))}e^{\alpha(f(\mathbf{x})-f(\mathbf{x}'))}\Big).\end{aligned} \tag{4.5}$$

Mallapragada *et al.* suggest the following approximations in order to simplify the minimization of the objective function. First, $e^{(\alpha(f_i-f_j))}$ is bounded as

$$e^{(\alpha(f_i-f_j))} \leq \frac{1}{2}(e^{2\alpha f_i} + e^{-2\alpha f_j}). \tag{4.6}$$

This results in an overall upper bound of the objective function

$$\begin{aligned} F \leq \sum_{\mathbf{x}' \in \mathcal{X}^U} \Big(& \sum_{(\mathbf{x},y)\in\mathcal{X}^L} s(\mathbf{x},\mathbf{x}')e^{-2y(F(\mathbf{x}')+\alpha f(\mathbf{x}'))} \\ & +\lambda_u \sum_{\mathbf{x}' \in \mathcal{X}^U} \frac{s(\mathbf{x},\mathbf{x}')}{2} e^{((F(\mathbf{x}')-F(\mathbf{x}))}\big(e^{2\alpha f(\mathbf{x})} + e^{-2\alpha f(\mathbf{x}')}\big)\Big). \end{aligned} \tag{4.7}$$

The objective function can be further written as

$$\overline{F} \leq \sum_{\mathbf{x}' \in \mathcal{X}^U} e^{(-2\alpha f(\mathbf{x}'))} p_i e^{(2\alpha f(\mathbf{x}')))} q_i. \tag{4.8}$$

p_i and q_i are two different terms depending on the label of y' and are formulated as

$$p_{\mathbf{x}} = \lambda_l \sum_{(\mathbf{x}',y')\in\mathcal{X}^L} \mathbb{I}(y'=1)s(\mathbf{x},\mathbf{x}')e^{-2F(\mathbf{x}')} + \frac{\lambda_u}{2} \sum_{\mathbf{x}\in\mathcal{X}^U} s(\mathbf{x},\mathbf{x}')e^{F(\mathbf{x}')-F(\mathbf{x})}, \tag{4.9}$$

and

$$q_{\mathbf{x}} = \lambda_l \sum_{(\mathbf{x}',y')\in\mathcal{X}^L} \mathbb{I}(y'=-1)s(\mathbf{x},\mathbf{x}')e^{-2F(\mathbf{x}')} + \frac{\lambda_u}{2} \sum_{\mathbf{x}\in\mathcal{X}^U} s(\mathbf{x},\mathbf{x}')e^{F(\mathbf{x})-F(\mathbf{x}')}, \tag{4.10}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Moreover, $p_{\mathbf{x}}$ and $q_{\mathbf{x}}$ can be considered as the confidence for a sample of being positive and negative, respectively. For details about the exact derivation of $p_{\mathbf{x}}$ and $q_{\mathbf{x}}$ we refer the reader to [Mallapragada et al., 2009]. Using these two terms, we compute the pseudo-labels and weights of unlabeled samples by:

$$\hat{y}_{\mathbf{x}} = \text{sign}(p_{\mathbf{x}} - q_{\mathbf{x}}) \tag{4.11}$$

and

$$w_{\mathbf{x}} = |p_{\mathbf{x}} - q_{\mathbf{x}}|. \tag{4.12}$$

Calculating the derivative of the loss function with respect to α and setting it to zero yields the optimal α for the weak classifier as

$$\alpha = \frac{1}{4} ln \frac{\sum_{\mathbf{x}\in\mathcal{X}^U} \big(p_i\mathbb{I}(f(\mathbf{x})=1) + q_i\mathbb{I}(f(\mathbf{x})=-1)\big)}{\sum_{\mathbf{x}\in\mathcal{X}^U} \big(p_i\mathbb{I}(f(\mathbf{x})=-1) + q_i\mathbb{I}(f(\mathbf{x})=1)\big)} \tag{4.13}$$

Therefore, at each step of boosting, we compute the pseudo-labels and weights of unlabeled samples and use them to find the best weak learner and its corresponding weight similar to the AdaBoost algorithm. Note that if no unlabeled data is used, the algorithm reduces to standard boosting. The detailed steps of the algorithm are summarized in Algorithm 4.1.

Algorithm 4.1 SemiBoost

Require: labeled training data $(\mathbf{x}, y) \in \mathcal{X}^L$ and unlabeled data $\mathbf{x}' \in \mathcal{X}^U$
Require: Similarity measure $s(\mathbf{x}, \mathbf{x}')$
Require: Weak learners f_i
Require: weight parameters λ_u, λ_l
Require: max iterations T
1: **for** $t = 1, 2, \ldots, T$ **do**
2: Compute p_i and q_i for every given sample
3: $\hat{y}_{\mathbf{x}} = \text{sign}(p_{\mathbf{x}} - q_{\mathbf{x}})$
4: $w_{\mathbf{x}} = |p_{\mathbf{x}} - q_{\mathbf{x}}|$
5: Train weak classifier $f_t(\mathbf{x})$
6: Compute α_t using Equation (4.13)
7: $F(\mathbf{x}) \leftarrow F(\mathbf{x}) + \alpha_t f_t(\mathbf{x})$
8: **end for**

4.2.2 Learning Visual Similarities

Being a manifold-based approach SemiBoost has the power to exploit both labeled and unlabeled samples if a similarity measure $s(\mathbf{x}, \mathbf{x}')$ is given. The similarity can be obtained from a distance measure $d(\mathbf{x}, \mathbf{x}')$, *e.g.*, by using a radial basis function

$$s(\mathbf{x}, \mathbf{x}') = e^{\left(-\frac{d(\mathbf{x},\mathbf{x}')^2}{\sigma^2}\right)}, \tag{4.14}$$

where σ^2 is the scale parameter [Zhu et al., 2003]. The crucial point is how to measure the distance $d(\mathbf{x}, \mathbf{x}')$ between points. Popular measurements are, for example, the Euclidean distance $||\mathbf{x} - \mathbf{x}'||$ or the Chi-Square distance $\frac{(\mathbf{x}-\mathbf{x}')^2}{2(\mathbf{x}+\mathbf{x}')}$.

As already mentioned above, a more powerful and flexible approach is to learn the distance function from labeled data, which is also known as metric learning. The advantage of discriminative learning of distance functions is that the metric can much better support task-specific classification. We use boosting to learn pair-wise distance functions similar to the method proposed by Hertz *et al.* [Hertz et al., 2004]. A distance function is a function of pairs of data points to be positive real numbers, usually (but not necessary)

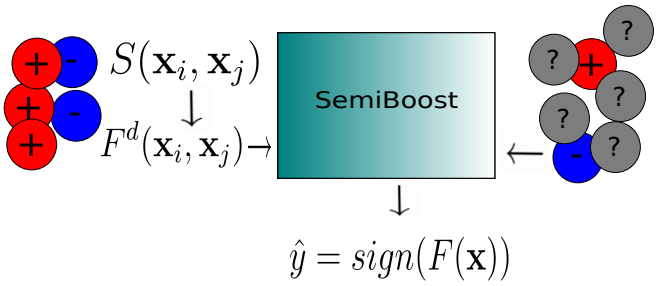


Figure 4.2: SemiBoost combined with a learned similarity measure from given labeled samples.

symmetric with respect to its arguments. The learning problem can be defined as a learning problem on the product space as $F^d : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Y} = [-1\ 1]$. To train a maximum margin classifier the training set

$$\begin{aligned} \mathcal{D}^d &= \{(\mathbf{x}, \mathbf{x}', +1) | y = y', \mathbf{x}, \mathbf{x}' \in \mathcal{D}^L\} \cup \\ &\quad \cup \{(\mathbf{x}, \mathbf{x}', -1) | y \neq y', \mathbf{x}, \mathbf{x}' \in \mathcal{D}^L\} \end{aligned} \tag{4.15}$$

is built by taking pairs of images of "same" and "different" class. Using pairs allows us to create a large number of training samples while having only a few labeled starting samples. In particular, if we omit pairs with self-similarities such as $(\mathbf{x}, \mathbf{x})$, we can create $\frac{n \cdot (n-1)}{2}$ training pairs out of n positive samples. The symmetry of the distance is not satisfied automatically, therefore it has to be enforced by introducing each pair twice, *i.e.*, both $(\mathbf{x}, \mathbf{x}')$ and $(\mathbf{x}', \mathbf{x})$. This also means that the number of training samples in fact becomes $n \cdot (n-1)$. Then, as in [Hertz et al., 2004] we use boosting to learn a classifier. The trained and normalized classifier $F^d(\mathbf{x}, \mathbf{x}') \in [-1\ 1]$ is interpreted as a distance

$$d(\mathbf{x}, \mathbf{x}') = \frac{1}{2}(F^d(\mathbf{x}, \mathbf{x}') + 1). \tag{4.16}$$

The conversion into a similarity $s(\mathbf{x}, \mathbf{x}')$ can be done as proposed in Equation (4.14), *i.e.*, using a radial basis function. This learned similarity can now be used as a prior for SemiBoost as depicted in Figure 4.2.

4.2.3 Using Arbitrary Classifiers as Similarity-Priors

Above, we have discussed the issue of training a pair-wise classifier which can serve as a distance measure for SemiBoost. However, sometimes it is the case that we have already

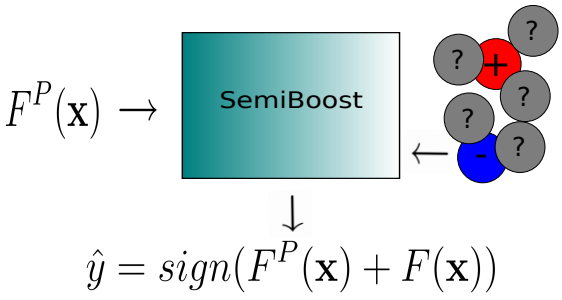


Figure 4.3: SemiBoost combined with a learned similarity measure from given labeled samples.

given a *non-pair-wise* classifier that is able to deliver confidence-rated predictions, for instance, an object detector, which can already (partially) solve our problem. We will denote such a classifier as *prior* or *prior classifier* $F^P(\mathbf{x})$. From a practical perspective and as can be seen in Figure 4.3, it would be now beneficial to incorporate such a prior into SemiBoost in order to exploit unlabeled data. Therefore, we have to approximate the pair-wise similarity $s(\mathbf{x}, \mathbf{x}')$ using the non-pair-wise prior $F^P(\mathbf{x})$.

In the following, we first show how the training can be done. Second, for evaluation we can use the prior by combining it with the newly trained classifier. Thereby, we benefit from the information which is already encoded in the prior classifier. Roughly speaking, the newly trained classifier can be rather "small", only correcting the mistakes of $F^P(\mathbf{x})$.

In more detail, we assume that we have access to a prior classifier $F^P(\mathbf{x}) : \mathcal{X} \rightarrow [-1, 1]$ (*e.g.*, an already trained face detector). The classifier has to provide a confidence measure of its classification. The more confident the decision is the higher the absolute value of the response. We can use boosting for training such a classifier and the responds can be translated into a probability [Friedman et al., 2000]. We can now incorporate this classifier into SemiBoost using the following approximation

$$|F(\mathbf{x}, \mathbf{x}')| \approx |F(\mathbf{x}) - F(\mathbf{x}')|. \tag{4.17}$$

In other words, a discriminative pair-wise function measuring the distance between $\mathbf{x}$ and $\mathbf{x}'$ is approximated by the difference of a conventional classification function for two samples.

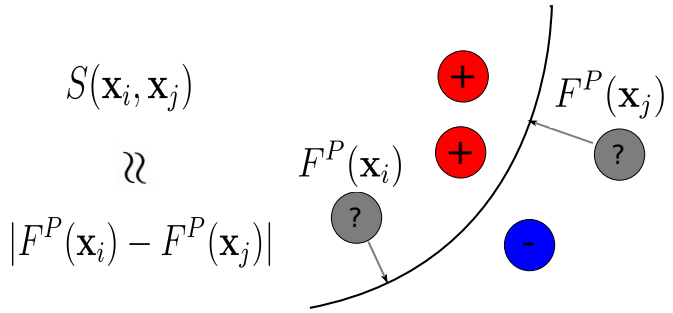


Figure 4.4: The similarity between two samples $\mathbf{x}$ and $\mathbf{x}'$ is approximated by the difference of the responses from an a priori given classifier $F^P(\cdot)$.

Discussion If $F(\mathbf{x}, \mathbf{x}')$ is a large margin function and two samples are identical $F(\mathbf{x}, \mathbf{x}')$ will be zero whereas $F(\mathbf{x}, \mathbf{x}')$ will be large the more dissimilar $\mathbf{x}$ and $\mathbf{x}'$ are. The same holds for taking $F(\mathbf{x}) - F(\mathbf{x}')$ which also will be zero if the two samples are identical and large the more dissimilar they are. Hence, approximating the similarity using a non-pairwise classifier corresponds to indirectly measuring the distance to the decision boarder. The principle is visualized in Figure 4.4.

According to this discussion, we define as distance measure

$$d(\mathbf{x}, \mathbf{x}') = |F^P(\mathbf{x}) - F^P(\mathbf{x}')| \tag{4.18}$$

as the absolute difference of the classifier response to the decision boundary. In other words, samples are similar if they have a similar classifier response. The distance is converted to a similarity using Equation (4.14) as described in the previous subsection. Now, we are able to proceed training on the proposed SemiBoost manner.

4.2.4 Classifier Combination

If we train a SemiBoost classifier $H(\mathbf{x})$ using the prior classifier $F^P(\mathbf{x})$ as similarity measure, it makes sense to use this prior knowledge for the final classification process as well (*i.e.*, combine the two classifiers). This is closely related to the approach proposed by Schapire *et al.* [Schapire et al., 2002]. Similarly, we use the prior knowledge as the 0^{th} weak classifier $f_0(\mathbf{x}) = \sigma^{-1}(P^P(y = 1|\mathbf{x}))$ where $P^P(y = 1|\mathbf{x})$ is the a priori probability of the sample corresponding to the positive class and $\sigma^{-1}(\cdot)$ is the inverse function of our logistic model (see [Friedman et al., 2000])). Since we use boosting to train the prior classifier, we end up with $f_0(\mathbf{x}) = F^P(\mathbf{x})$ which is included in the combined classifier $F^C(\mathbf{x}) = \alpha_0 F^P(\mathbf{x}) + F(\mathbf{x})$. Note that for ease of notation, in the following we will write $F^P(\mathbf{x}) = \alpha_0 F^P(\mathbf{x})$.

Similar to the standard boosting we can take a look at the expected value of the loss function [Friedman et al., 2000] and compared to Equation (2.10) we get for the combined classifier

$$P(y=1|\mathbf{x}) = \frac{e^{F^P(\mathbf{x})+F(\mathbf{x})}}{e^{F^P(\mathbf{x})+F(\mathbf{x})} + e^{-F^P(\mathbf{x})-F(\mathbf{x})}}. \tag{4.19}$$

If we are only interested in the decision we see that a sample is classified as positive if we set $P(y=1|\mathbf{x}) \geq 0.5$ and after some mathematical rewriting we get

$$\hat{y} = \operatorname{sign}\left(\sinh(F^P(\mathbf{x}) + F(\mathbf{x}))\right) = \operatorname{sign}\left(F^P(\mathbf{x}) + F(\mathbf{x})\right). \tag{4.20}$$

Discussion The interpretation is as follows. A label switch can happen, *i.e.*, $F(\mathbf{x})$ can overrule $F^P(\mathbf{x})$, if the combined term has a different label as the prior $F^P(\mathbf{x})$. As can be easily seen, this is the case if $|F| > |F^P|$. Therefore, the more confident the prior is, the harder it is that the label changes. We do not make any statements whether this is a correct or incorrect label switch. Note that overall the prior classifier can be wrong, but it has to provide an "honest" decision. Meaning, if it is highly confident it must be ensured to be a correct decision. There are also relations to the co-training [Balcan et al., 2004] assumptions, *i.e.*, a classifier should be never "confident but wrong". By rewriting Equation (4.20) as

$$\begin{aligned}\hat{y} =& \operatorname{sign}(\sinh(F^P(\mathbf{x}) + F(\mathbf{x})) = \\ =& \operatorname{sign}(\cosh(F(\mathbf{x}))\sinh(F^P(\mathbf{x})) + \cosh(F^P(\mathbf{x}))\sinh(F))\end{aligned} \tag{4.21}$$

one sees that the two classifiers weight each other using hyperbolic functions. The factor obtained by $\cosh(\cdot) \geq 1$ weights the decision of the corresponding classifier passed through the asymmetric $\sinh(\cdot)$ function. By an additional scaling factor more emphasis can be put either on the prior or the newly trained classifier; however, this is not explored in this thesis.

Summarizing, after training $F(\mathbf{x})$ the expected target of an example is obtained by a combined decision. The combined classifier can now be interpreted as improving $F^P(\mathbf{x})$ using labeled and unlabeled samples. We train $F(\mathbf{x})$ with SemiBoost using labeled and unlabeled data, since $F^P(\mathbf{x})$ is used to calculate the similarity via (Equation (4.18) and Equation (4.14)) these two classifiers are tightly coupled via the training process and Equation (4.21) is not just a simple sum rule. If we use a complex classifier, *i.e.*, consisting of many weak classifiers, and have a lot of training data $F(\mathbf{x})$ will "absorb" the entire knowledge of $F^P(\mathbf{x})$; therefore the usual setting is that we us a rather small $F(\mathbf{x})$ to only correct $F^P(\mathbf{x})$.

4.2.5 Toy Experiment

In the following, we will illustrate the basic learning behavior of the algorithm using a simple toy experiment. We consider a two class classification problem depicted in Figure 4.5. The underlying data generating process produces positive samples around the point $(0.5, 0.5)$ and negative samples at a circle centered at the same point with radius 1 (both with variance 0.1). First, we train a "common" boosting classifier (as weak classifiers a linear separator is used) on just the labeled examples (red and blue circles). Second, we use our proposed SemiBoost approach. Additionally, we use 100 unlabeled points (black crosses) drawn from the distributions above. As distance measure the Euclidean distance is used and converted to a similarity measure using Equation (4.14) with $\sigma^2 = 0.01$. The left side of the plot shows the samples and the decision border. The right side of each subfigure depicts the probability for the positive class $P(y = 1|\mathbf{x})$. Figure 4.5(a) shows a weak decision due to the limited number of samples, Figure 4.5(b) using additional unlabeled data an essentially improved decision is obtained by SemiBoost.

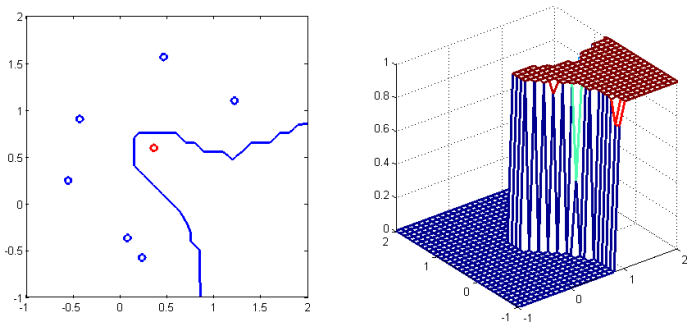
The second toy example (Figure 4.6) shows improvement of a prior classifier. We build an "honest" prior by estimating the positive and negative probability using a kernel density estimation (Gaussian-distribution with $\sigma^2 = 0.05$) on the labeled samples. This prior serves as similarity measures for SemiBoost, which is used to train a small classifier (10 weak classifiers). The combined classifier performs better than the prior and the newly trained alone, respectively.

4.3 Experiments on Visual Classification and Detection

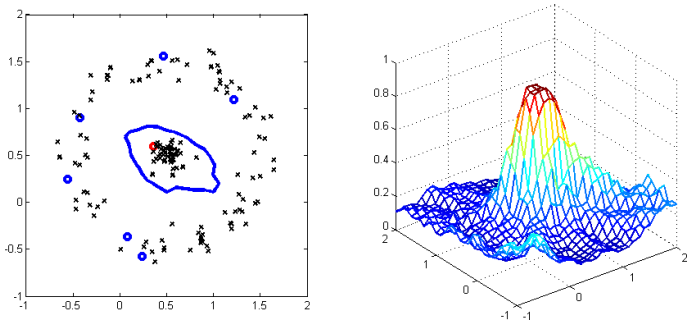
The goal of the following experiment is to demonstrate the applicability of the proposed method on visual object classification and detection. In particular, we want to show that SemiBoost combined with learning of visual similarities can be used in various applications without needing additional information such as background modeling or high-level features.

Classification from Few Labeled Examples

For the first experiment, we collected a set of 1100 30×30 car patches with the help of a simple motion detector from a common traffic scene (see Figure 4.7) and, additionally, 1100 random negative patches from the same scene. 300 positives and 300 negatives were kept as an independent test set. In order to train the classifiers we use simple Haar-like features as in [Viola and Jones, 2001].



(a) Boosting with labeled data only



(b) SemiBoost: Boosting with labeled and unlabeled data

Figure 4.5: Toy Example 1: Positive and negative labeled (red and blue circles) and unlabeled samples (back crosses) are used for learning via "common" boosting (a) and using the proposed SemiBoosting approach (b) which additionally takes unlabeled data into account.

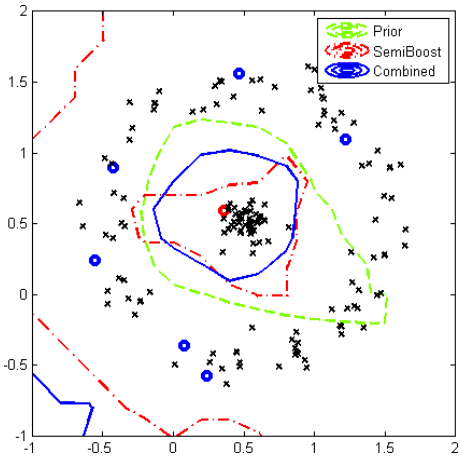


Figure 4.6: Toy Example 2: The decision boundary of an "honest" prior (green) is "corrected" by a SemiBoost classifier (red) and the combined decision boundary (blue) is archived.

(a) Car samples (b) Background samples

Figure 4.7: Some representative collected positive and negative input samples.

First, we trained a similarity function on randomly selected pairs of images as explained in Section 4.2.2. The similarity function was evaluated by converting the normalized classifier $F(\mathbf{x}, \mathbf{x}')$ to a distance measure using Equation (4.16) and passing it through a radial basis function (Equation (4.14)) with $\sigma = 0.01$. Then, we measured the average distance similarity to the labeled positive and labeled negative samples, respectively. This mean similarity was then converted to a hard label depending on whether the test sample was on average more closely to negatives or positives, *i.e.*,

$$C(\mathbf{x}) = \underset{k \in \{-1,1\}}{\arg\max} \frac{1}{|\mathcal{X}_k|} \sum_{\mathbf{x}'} s(\mathbf{x}, \mathbf{x}'). \tag{4.22}$$

In Figure 4.8 we depict the classification performance of the similarity measure over the boosting iterations on the unlabeled data. We varied the number of training samples from 5 to 20. As can be observed, even with a small amount of labeled samples very good classification accuracies can be achieved. However, this method has the disadvantage that (i) all the labeled samples have to be kept for evaluation and (ii) in order to get a prediction for a test sample, its distance has to be measured to all the training samples, which is computationally very expensive. Therefore, in the following we will train a classifier $F(\mathbf{x})$ which during evaluation does not have these disadvantages of similarity functions but rather uses a trained pair-wise similarity function during training in order to exploit unlabeled data.

In the next experiment, we trained a similarity classifier with 15 labeled samples and used it as prior for SemiBoost. In Figure 4.9(a) we depict the performance measured in terms of receiver-operator characteristic (ROC) curve of the final classifier depending on how many unlabeled samples were used. As can be seen, the more unlabeled samples are

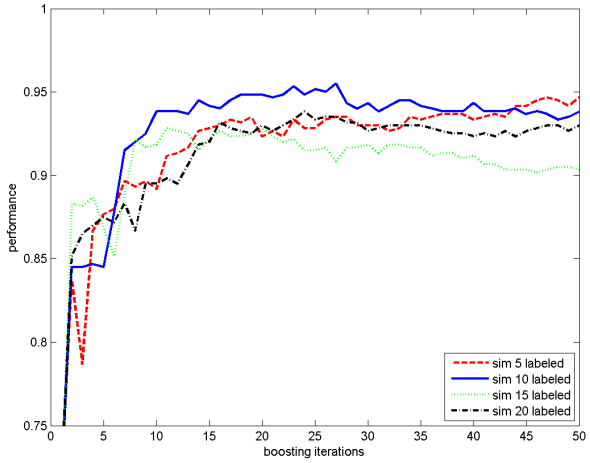
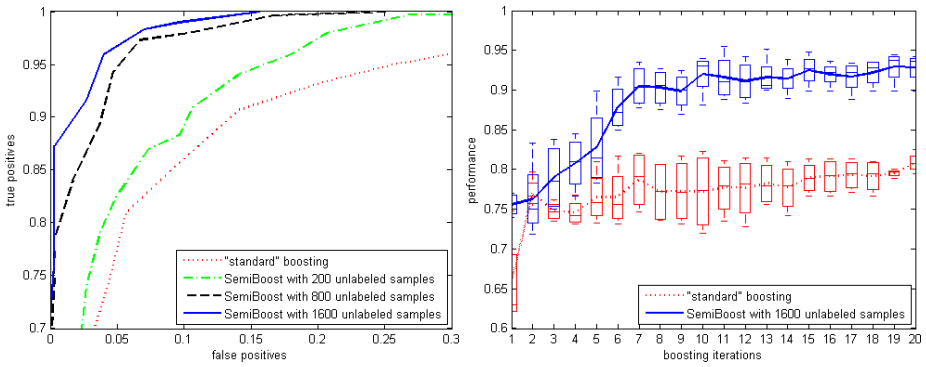


Figure 4.8: Performance of the similarity over the iterations.



(a) ROC depending on the number of unlabeled samples

(b) classifier improvement over the boosting iterations

Figure 4.9: Learning of a car-detector: Performance of the proposed approach improves significantly compared to the common approach (no unlabeled data is used) both when (a) including more weak classifiers and (b) use more unlabeled data.

exploited, the better is the final performance. In Figure 4.9(b) we show how the performance increases for 1600 unlabeled samples over the boosting iterations in comparison to standard AdaBoost. Note that the final boxplot was obtained by repeating the experiment 5 times. As can be seen, the performance increases continuously when adding further weak classifiers and significantly outperforms the standard boosting.

Improving a Detector

Contrary to the previous experiment, where we used a pair-wise classifier, in this experiment we want to show that with our method it is possible to improve any given classifier on related unlabeled samples as proposed in Section 4.2.4. Therefore, for each of the following two experiments, we first train a Viola/Jones detector [Viola and Jones, 2001] on labeled data. We denote this classifier as $F^P(\mathbf{x})$. The response of the last cascade layer is used as our prior classifier. The final detection results are obtained by non-maxima suppression as post processing step.

Figure 4.10(a) depicts the results by applying the prior classifier trained on the frequently used MIT+CMU faces where state-of-the-art results are achieved. Now, we want to improve this classifier using related unlabeled data; *i.e.*, the unlabeled data set should contain a significant amount of faces. Since this also can be a tedious task, we propose a simple approach to data-mine related unlabeled samples using web image search engines, where the key idea is to feed the search engine with queries that might lead to resulting images containing our target objects. For instance, if we want to train a car detector, we could use "highway", "road", "traffic", *etc.* as queries and denote the obtained images as our first unlabeled data set $\mathcal{X}^U$. Of course, the thus obtained images might be still very noisy. So, in a next step we refine $\mathcal{X}^U$ by applying $F^P(\mathbf{x})$ in a sliding-window manner over different scales in order to bootstrap for "interesting" unlabeled samples. We crop all detected objects of $F^P(\mathbf{x})$ and copy them to a new unlabeled set $\mathcal{X}^{U^*}$. After having obtained $\mathcal{X}^{U^*}$ and some labeled data $\mathcal{X}^L$, we can now apply SemiBoost in order to improve $F^P(\mathbf{x})$. We depict the approach in Algorithm 4.2.

Algorithm 4.2 Simple data mining for informative unlabeled data

Require: Labeled training data $(\mathbf{x}, y) \in \mathcal{X}^L$

1: Train cascaded detector $F^P(\mathbf{x})$ on $\mathcal{X}^L$ using [Viola and Jones, 2001]
2: Use a web image search engine in order to collect huge amounts of possibly useful images $\mathcal{X}^U$; pass phrases that are much likely related to your target object
3: Apply $F^P(\mathbf{x})$ in a sliding window manner on $\mathcal{X}^U$ and copy all detections to $\mathcal{X}^{U^*}$
4: Train a SemiBoost classifier $F(\mathbf{x})$ on $\mathcal{X}^L$ and $\mathcal{X}^{U^*}$ using $F^P(\mathbf{x})$ as prior
5: Output the final classifier $F(\mathbf{x})$

As proof of concept, we applied $F^P(\mathbf{x})$ on 300 random images downloaded from Google-Image search with the keyword "team". The delivered detections (>4000) are used as additional unlabeled data. The 50 most confident detections were used as positive labeled data and the 50 least confident detections were used as negative ones for training the SemiBoost classifier with only 30 weak classifiers. The proposed combination strategy (Equation (4.20)) improved the results (higher detection and lower false positive rate

(a) prior (b) trained (c) combined

Figure 4.10: Detection results of a face detector (a) which serves as prior to build a SemiBoost classifier using additional unlabeled data. This classifier alone (b) has not the power of delivering good results but the combination improves the result essentially.

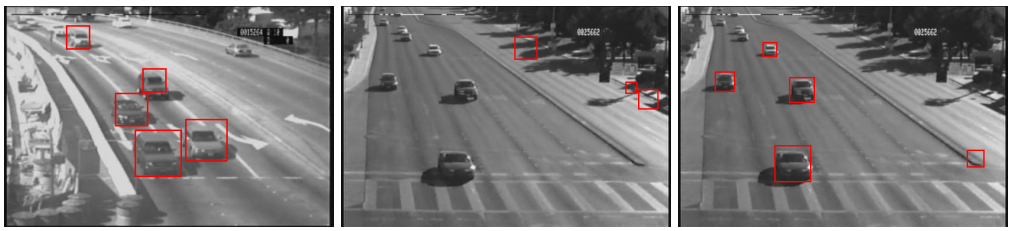
as well as a better alignment of the detections) as shown in Figure 4.11. Note, the trained classifier alone consists only of 30 weak classifiers which yields poor results when applied on the image (Figure 4.10). Of course, when using more weak classifiers it will learn the prior information as well. Additionally, Figure 4.11 shows representative examples which were obtained by our approach. Note that the most related approach, *i.e.*, [Levin et al., 2003], fails on this task when only using single views and we hence omit the detailed comparison.

Scene Adaption

In the next experiment, we want to test our approach on a typical task occurring often in practice; *i.e.*, scene adaptation or visual transfer learning. In particular, in multi-camera surveillance scenarios instead of collecting huge amounts of labeled data in order to train a general object detector one can collect data for specific camera views or scenarios and train a classifier only on these. This has the advantage that the model complexity can be reduced, which results in faster detectors, and that usually the accuracy can be increased because the classifier has only to discriminate the target object versus one specific background. However, as collecting and labeling the data for each camera is cost and time intensive (*e.g.*, consider networks with tens or hundreds of surveillance cameras), one is interesting in reusing as much information as possible between the cameras for training. In the following, we demonstrate that our approach is also useful for such a problem.

For this purpose, we trained a car detector for a specific scene (*i.e.*, "scene 1" illustrated in Figure 4.12(a)) using 1000 labeled samples. When applying this classifier on a different scene with a similar view point, as expected, it performs significantly worse. Hence, in order to adapt the detector to the different scene (*i.e.*, "scene 2", Figure 4.12(b)), we apply a simple motion detector to get potential positive samples of scene given in Figure 4.12(b). After collecting 1000 of them and additionally cropping 1000 random

Figure 4.11: Detection results of a state-of-the-art face detector (left) and the improved results obtained by the proposed strategy (right). As can be seen, incorporating additional unlabeled data helps to increase both recall and precision.



(a) scene 1 (b) scene 2 (c) improved results

Figure 4.12: A scene specific car detector for scene 1 (a) is applied on a "similar" scene (b). The poor behavior can be significantly improved using unlabeled data taken from the second scene as shown by a typical frame (c).

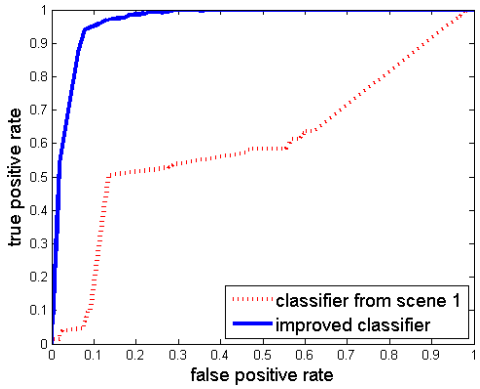


Figure 4.13: ROC curve for "scene 2" for the starting classifier (red) and the improved classifier (blue) using unlabeled samples from the target scene.

sub-patches from the scene these 2000 serve as unlabeled examples to train a SemiBoost classifier with 30 weak classifiers using the classifier trained on scene 1 as prior. A typical frame superimposed with the detection result is shown in Figure 4.12(b)(c). The detection results improved (much lower false positive rate and higher detection rate) as shown in Figure 4.13.

4.4 Summary

In this chapter, we introduced a boosting algorithm that combines semi-supervised learning with learning of distance metrics. Learning distance metrics is especially interesting for computer vision applications because for a given task it is often hard to say which metric to take. Distance functions can be learned using a small amount of data and can be used as prior information in order to support the exploitation of unlabeled data during a SSL process. We demonstrated the approach on several vision applications such face detection and visual transfer learning. In the following chapter, we will show how to extend

this approach to on-line learning.

Chapter 5

On-line Semi-Supervised Boosting

Although there have been proposed a large amount of approaches to the SSL task, most of them operate in off-line or batch mode [1]. Recall from Chapter 2 that off-line methods assume having access to the entire training data at any time, which eases optimization and typically yields good classifiers. In practice, however, on-line learning capability is demanded because learners often have limited access to the problem domain due to dynamic environments or streaming data sources. Additionally, on-line learning methods usually require less memory to operate which makes them ideal for incremental learning problems, interactive learning or in cases where limited hardware resources do not allow for full data storage, for instance on mobile devices. Finally, in supervised learning on-line methods are frequently applied to large scale data problems due to their inherent good scaling characteristics.

These benefits make on-line approaches also interesting for semi-supervised learning and leverage the usage of SSL in scenarios that require sequential data analysis. Furthermore, the goal of SSL is to benefit from large amounts of unlabeled data. Yet, the bad scaling behavior of current state-of-the-art SSL methods, which can be up to $\mathcal{O}(n^3)$ [Mann and Mccallum, 2007], where n is the number of unlabeled samples, often limits their applicability in practice. This seems to be in particular paradox in a field that claims to benefit from the huge amounts of unlabeled data that are often available for free. Hence, incorporating on-line learning techniques into SSL has also the potential to make them operate in really large data scenarios. Although on-line learning can bring several advantages to semi-supervised learning, in the literature there exist only a small amount of approaches dealing with on-line SSL, *e.g.*, methods based on deep neural networks [Weston et al., 2008] or convex programming in kernel space [Goldberg et al., 2008].

In this chapter, we discuss on-line semi-supervised learning algorithms based on on-line boosting [Oza, 2001]. The reasons for developing such algorithms are that boosting

[1] Note that throughout this text we use the two terms interchangeably with equal meaning.

is one of the most successful learning algorithms and its on-line variant is easy to implement, practically converges to the off-line version [Oza, 2001] and is frequently applied in practice. Additionally, as we already shortly discussed in Chapter 1, in this work, we target visual object tracking based on "tracking-by-detection" as one of our main applications and on-line boosting has shown to be a very powerful and fast method for this task [Grabner and Bischof, 2006]. As we will discuss later in more detail, *i.e.*, in Chapter 9, visual object tracking based on on-line supervised learning suffers severely from the so called "drifting problem"; *i.e.*, due to the self-learning nature of this approach small tracking errors, such as misalignment, wrong updates, *etc.*, can accumulate over time and finally lead to failure of the approach. One explanation for this problem is that usually supervised learners are applied for the tracking task; *i.e.*, the learning method assumes all samples being truely labeled, in the ideal case without class-label noise. However, in the tracking application, labeled data can only be provided at the first frame when the target object is usually marked by hand. During the tracking process, the classifier has to label the samples (patches) by itself; it thus has to perform self-learning. Since in self-learning scenarios noisy labels are quite common, applying a supervised learning method is the main cause for "drifting". This argument is even more valid for on-line boosting which due to the exponential loss function is known to be highly susceptible to class label noise [Long and Servedio, 2008b]. Consequently, we highlight that tracking-by-detection can naturally be considered as a semi-supervised learning problem; *i.e.*, a small fraction of labeled data is only available at the very first frame while in all subsequent frames during on-line operation no true labels can be considered to be available. Thus, for tracking the ideal learner is a semi-supervised method, which inherently is able to cope with both labeled and unlabeled data.

5.1 On-line SemiBoost

On-line boosting for feature selection, as described above, is a fully supervised learning algorithm. In the following, we show how to extend on-line boosting to the semi-supervised case using SemiBoost introduced in Chapter 4.

For our on-line variant of SemiBoost we adopt the combined loss function given in Equation (4.3). For minimizing this loss we use on-line boosting for feature selection as described above. In order to get weight and label estimates for unlabeled samples, according to the formulation of SemiBoost, we have to calculate $p_{\mathbf{x}}$ and $q_{\mathbf{x}}$ defined in Eq. 4.9 and Equation 4.10, respectively. Yet, these terms cannot be evaluated directly due to the sums over pairs of either labeled and unlabeled samples. Since we are in a pure on-line setting we cannot access the whole training data at once. Hence, in the following we show how to circumvent the pair-wise calculations in the on-line setting using several

approximations.

First, we need a similarity measurement $s(\mathbf{x}, \mathbf{x}')$ in order to incorporate the unlabeled data. As we mentioned in the previous chapter, in principle, $s(\mathbf{x}, \mathbf{x}')$ can be any similarity function, also a matrix where the distances among the samples are encoded. However, since in the on-line setting it is not possible to calculate such a matrix a priori, we learn the similarity $s(\mathbf{x}, \mathbf{x}') \approx F^{sim}(\mathbf{x}, \mathbf{x}')$ by a classifier using boosting similar to [Hertz et al., 2004]. Furthermore, we only have to sum over the similarity of the current (unlabeled) sample $\mathbf{x}$ and the set of positive or negative samples. Given the labeled samples in advance, we can train a classifier a priori which measures the similarity, to the positive or negative class. If one of the samples is always positive this can be approximated by learning a classifier which has to describe the positive class $\sum_{\mathbf{x}_i \in \mathcal{X}^+} F^{sim}(\mathbf{x}, \mathbf{x}_i) \approx F^+(\mathbf{x})$, *i.e.*, provides a probability measure that $\mathbf{x}$ corresponds to the positive class. In the same manner, a classifier is built for the negative class $\sum_{\mathbf{x}_i \in \mathcal{X}^-} F^{sim}(\mathbf{x}, \mathbf{x}_i) \approx F^-(\mathbf{x})$. Instead of learning generative classifiers, we propose to learn a discriminative classifier $F^P(\mathbf{x})$ which has to distinguish between the two classes, *i.e.*, $F^+(\mathbf{x}) \sim F^P(\mathbf{x})$ and $F^-(\mathbf{x}) \sim 1 - F^P(\mathbf{x})$. Since we use boosting to learn such a prior classifier, it can be translated into a probability using $p(y = 1|\mathbf{x}) = (1 + e^{-H(\mathbf{x})})^{-1}$.

Plugging this observations into the former definitions we get

$$\tilde{p}_{\mathbf{x}} \approx e^{-F_{n-1}(\mathbf{x})} \sum_{\mathbf{x}_i \in \mathcal{X}^+} S(\mathbf{x}, \mathbf{x}_i) \approx e^{-F_{n-1}(\mathbf{x})} F^+(\mathbf{x}) \approx \frac{e^{-F_{n-1}(\mathbf{x})} e^{F^P(\mathbf{x})}}{e^{F^P(\mathbf{x})} + e^{-F^P(\mathbf{x})}}, \tag{5.1}$$

$$\tilde{q}_{\mathbf{x}} \approx e^{F_{n-1}(\mathbf{x})} \sum_{\mathbf{x}_i \in \mathcal{X}^-} S(\mathbf{x}, \mathbf{x}_i) \approx e^{F_{n-1}(\mathbf{x})} F^-(\mathbf{x}) \approx \frac{e^{F_{n-1}(\mathbf{x})} e^{-F^P(\mathbf{x})}}{e^{F^P(\mathbf{x})} + e^{-F^P(\mathbf{x})}}. \tag{5.2}$$

Since we are interested in the difference of the approximated values, we finally get the "pseudo-soft-label"

$$\tilde{z}_n(\mathbf{x}) = \tilde{p}_n(\mathbf{x}) - \tilde{q}_n(\mathbf{x}) = \frac{\sinh(F^P(\mathbf{x}) - F_{n-1})}{\cosh(F^P(\mathbf{x}))} = \tanh(F^P(\mathbf{x})) - \tanh(F_{n-1}(\mathbf{x})). \tag{5.3}$$

It is now straight forward to extend SemiBoost to on-line boosting. As already stated above, for the labeled examples $(\mathbf{x}_n, y_n), y \in \{-1, +1\}$ nothing changes. For each unlabeled sample $\mathbf{x}_n$ while propagating through the selectors, after each selector not only the weight (the importance λ_n) of the example is adapted, but also its target y_n may change. Hence, for unlabeled samples in each iteration n, we set

$$\tilde{y}_n = \text{sign}(\tilde{z}_n(\mathbf{x})) \text{ and } \lambda_n = |\tilde{z}_n(\mathbf{x})|, \tag{5.4}$$

where $\tilde{z}_n(\mathbf{x})$ is defined in Equation 5.3.

Discussion As we have seen, by training a prior classifier F^P from labeled samples a priori provided, it is possible to include unlabeled data into the on-line boosting framework using pseudo-labels and pseudo-importances. Note that if some labeled data is given in advance, this data can be used to train the prior (or even update the prior when it arrives over time). However, it should also be clear that the incorporation of prior knowledge is not limited to a prior classifier and in principle any source of prior knowledge can be incorporated. Our on-line semi-supervised boosting algorithm for feature selection is sketched in Algorithm 5.1. As can be seen, compared to the original on-line boosting algorithm [Grabner and Bischof, 2006] only a few lines of code have to be changed in order to incorporate unlabeled data.

5.2 On Robustness of On-line Boosting

Currently, boosting is one of the best and thus one of the mostly applied classification methods in machine learning. This is also true for the on-line variant, which is frequently applied in practice. However, boosting is proven, from both the theoretical and the experimental point of view to be sensitive to label noise. For off-line boosting, this issue was discovered relatively early and, hence, different more robust methods [Maclin and Opitz, 1997, Mason et al., 1999, Dietterich, 1998, Friedman et al., 2000, Freund, 2000, Domingo and Watanabe, 2000, Long and Servedio, 2008b, Masnadi-Shirazi and Vasconcelos, 2008] have been proposed.

Although for on-line boosting robustness is highly important because it is frequently applied in autonomous learning problems, in contrast to the off-line case, up to now this issue has not been studied. Thus, in the following, we study the robustness of on-line boosting for feature selection in terms of label noise. In particular, we follow the work of Long *et al.* [Long and Servedio, 2008b], who showed that the loss function has not only high influence to the learning behavior but also on the robustness. Especially convex loss functions (typically used in boosting) are highly susceptible to random noise. Hence, to increase the robustness the goal is to find more suitable less noise sensitive loss functions. For that purpose, we first introduce a generic boosting algorithm, which we call *On-line GradientBoost*, where arbitrary loss functions can be plugged in easily. In fact, this method extends the GradientBoost algorithm of Friedman *et al.* [Friedman, 2001] and is similar to the AnyBoost algorithm of Mason *et al.* [Mason et al., 1999]. Based on this algorithm, we develop different on-line boosting methods using the loss functions proposed for robust off-line boosting algorithms.

We talk about label noise, if a sample was assigned a wrong label during the labeling process of the data. AdaBoost is highly susceptible to noise. This sensitivity comes from the fact that AdaBoost increases the weight of a mis-classified sample in each iteration.

Algorithm 5.1 On-line SemiBoost

Require: A training sample: $(x_n, y_n) \in \mathcal{X}_L$ or $(x_n) \in \mathcal{X}_U$.
Require: Prior knowledge: F^P
Require: Number of selectors M.
Require: Number of weak learners per selector K.

Set the initial weight $\lambda_m = e^{-yF(0)} = 1$
for $m = 1$ to M **do**
    // Update weight estimation
    if $(x_n, y_n) \in \mathcal{X}_L$ **then**
        $y_m = y, \lambda_m = e^{-yF_{m-1}(\mathbf{x})}$
    else
        // Update pseudo label and weight
        $\tilde{z}_m(\mathbf{x}) = \tanh(F^P(\mathbf{x})) - \tanh(F_{m-1}(\mathbf{x}))$
        $y_m = sign(\tilde{z}_m(\mathbf{x})), \lambda_m = |\tilde{z}_m(\mathbf{x})|$
    end if
    for $k = 1$ to K **do**
        Train k^{th} weak learner $f_m^k(x)$ with sample (x_n, y_m) and weight λ_m.
        Estimate the error:
        if $f_{m,k}^{weak}(\mathbf{x}) == y$ **then**
            $\lambda_{m,k}^c = \lambda_{m,k}^c + \lambda_k$
        else
            $\lambda_{m,k}^w = \lambda_{m,k}^w + \lambda_k$
        end if
        $e_m^k = \frac{\lambda_{n,m}^w}{\lambda_{n,m}^c + \lambda_{n,m}^w}$
    end for
    Find the best weak learner with the least total weighted error: $j = \arg\min_k e_m^k$.
    Set $f_m(x_n) = f_m^j(x_n)$.
    Set $\alpha_m = \frac{1}{2} \ln \left(\frac{1-e_m}{e_m} \right)$
end for
Output the final model: $F(x)$

This re-weighting strategy allows boosting to concentrate on hard samples while easy samples are less emphasized. However, if the sample has a wrong label and the previous weak learners are assigning the true (but hidden) label to the sample, AdaBoost still will consider this as a mis-classification and dramatically (exponentially) increase its weight. This can finally corrupt the learning result. Therefore, the performance of the boosting algorithm will be highly dependent on the presence of such noisy samples.

In the case of off-line boosting, Maclin and Optiz [Maclin and Opitz, 1997] were one of the first to note the sensitivity of the AdaBoost algorithm [Freund and Schapire, 1996] to label noise. Later, Dietterich [Dietterich, 1998] conducted more experimental studies analyzing different ensemble building methods and also noted the sensitivity of AdaBoost to label noise. Mason *et al.* [Mason et al., 1999] was one of the first to analyze boosting algorithms in the context of functional gradient descent. They also proposed a theoretically inspired loss-function and a boosting algorithm using that loss called *DoomII*, which showed increased robustness to label noise. Next, in their seminal work Friedman *et al.* [Friedman et al., 2000] showed the connection of boosting algorithms to the stagewise additive logistic regression methods from the applied statistics domain. Based on minimizing the negative log-likelihood, they proposed a loss-function and a corresponding boosting algorithm called *LogitBoost*, which also showed to be more resistant to label noise. Domingo and Watanabe [Domingo and Watanabe, 2000] and Freund [Freund, 2000] also investigated this issue by proposing the *MadaBoost* and *BrownBoost* methods, respectively, which also try to decrease the label noise sensitivity of AdaBoost.

Random Noise Defeats all convex potential Boosters Fast forwarding to recent works in this field, Long and Servedio [Long and Servedio, 2008b] analyzed different convex loss-functions used in designing boosting algorithms and showed that from a theoretical point of view all of these methods will be sensitive to the label noise. In more detail, consider any boosting algorithm using a non-convex loss function. In case of no noise, there exists a simple data set which is easily learnable by this boosting algorithm. However, if there exists a nonzero random classification noise rate ν, the same data set cannot be learned more accurately than $\frac{1}{2}$.

Definition 1. *A function $f : \mathcal{R} \rightarrow \mathcal{R}$ is a convex potential function if it satisfies the following properties:*

1. *f is convex and nonincreasing*
2. *f is differentiable and f' is continuous*
3. *$f'(0) < 0$ and $\lim_{x\rightarrow\infty} f(\boldsymbol{x}) = 0$*

Many well known boosting algorithms are based on potential functions f that satisfy the above Definition 1 and have thus problems with noise. Among these algorithms are AdaBoost [Freund and Schapire, 1999], LogitBoost [Friedman et al., 2000] and MadaBoost [Domingo and Watanabe, 2000]. For a detailed analysis and proof we refer the reader to [Long and Servedio, 2008b].

However, we would like to highlight that there exist efficient boosting algorithms that are based on non-convex loss functions such as Boosting by Majority [Freund, 1995],

BrownBoost [Freund, 2000] or MartingaleBoost [Long and Servedio, 2005, Long and Servedio, 2008a]. Also recently, Masnadi-Shirazi and Vasconcelos [Masnadi-Shirazi and Vasconcelos, 2008] studied the problem of loss-function design from the perspective of *probability elicitation* in statistics and, based on this, derived a non-convex loss-function for boosting. This algorithm, denoted as *SavageBoost*, has shown to be highly resistant to the presence of label noise while also converging fast in case of no noise.

5.2.1 Loss-Functions

By reviewing robust off-line boosting algorithms in the previous section, we can realize that most of the efforts in order to remedy the noise sensitivity weakness of boosting has been mainly focused on either heuristics or designing better loss-functions. However, increased noise robustness based on heuristics has often the drawback the algorithm performs worse compared to the original AdaBoost in situations without any label noise, whereas better loss-function design has shown to yield algorithms that are more noise robust but simultaneously do not perform worse compared to the original method in case of no noise. Hence, in the following, we will concentrate on how to leverage the usage of more robust loss functions for on-line boosting.

In Figure 5.1 we depict a few of the loss-functions commonly used in boosting and other machine learning methods. The corresponding loss-functions used in this figure are shown in Table 5.1. Here, $y \in \{-1, +1\}$ are the binary labels of a sample x and $F(x)$ is the real output of the classifier with the decision rule of $\hat{y} = \text{sign}(F(x))$. Traditionally, $yF(x)$ is called the *classification margin* of a sample. For the binary case, *i.e.*, $y \in \{-1, +1\}$ the margin is defined as

$$m(\mathbf{x}, y; \mathbf{f}) = f_y(\mathbf{x}) - f_{k \neq y}(\mathbf{x}) \tag{5.5}$$

It can easy be seen that the margin should be positive, *i.e.*, $m(\mathbf{x}, y; \mathbf{f})$ for a correct classification and negative in case of mis-classification, respectively. Based on the margin, different boosting algorithms apply different loss functions $\ell(\mathbf{x}, y; \mathbf{f})$ that usually upper bound the zero-one loss, *i.e.*, $\ell_{0-1}(\mathbf{x}, y; \mathbf{f}) = \mathbb{I}(m(\mathbf{x}, y; \mathbf{f}) \leq 0)$ but are easier to optimize.

From Figure 5.1 it is clear that the main difference between these loss functions is how they deal with mis-classified samples. There are two scenarios for mis-classification of a sample:

1. The sample is noise-free and it is the learning model which is not able to classify it correctly.

2. The sample has a label noise and the learning model is recovering its true (but hidden) label.

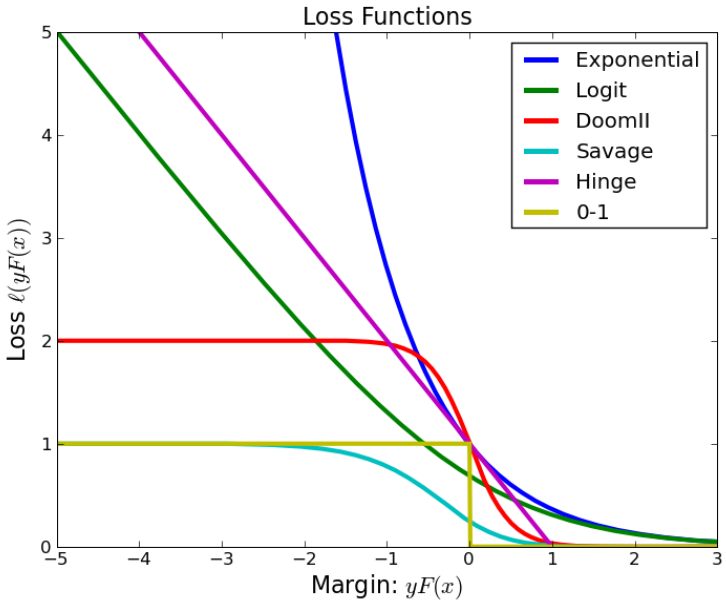


Figure 5.1: Different loss functions used in boosting and supervised machine learning methods.

Losses	Functions
0-1 Loss	$\ell_{0-1}(yF(x)) = \mathbb{I}(yF(x) < 0)$
Exponential [Freund and Schapire, 1996]	$\ell_{exp}(yF(x)) = \exp(-yF(x))$
Logit [Friedman et al., 2000]	$\ell_{log}(yF(x)) = \log(1 + \exp(-yF(x)))$
DoomII [Mason et al., 1999]	$\ell_{doom}(yF(x)) = 1 - \tanh(yF(x))$
Savage [Masnadi-Shirazi and Vasconcelos, 2008]	$\ell_{sav}(yF(x)) = 1/(1 + \exp(2yF(x)))^2$
Hinge	$\ell_{hin}(yF(x)) = \max(0, 1 - yF(x))$

Table 5.1: Loss functions used in Figure 5.1.

More specifically for a binary problem, let y^n be a noisy label, *i.e.*, $y^n = -y^t$, where y^t is the true label. If $y = y^t \neq \hat{y}$, we have to consider the first case; if $y = y^n \neq \hat{y}$ (and equally $\hat{y} = y^t$), the second one. For both cases, the higher the confidence of the classifier $F(x)$ the more the margin will be located towards the left part of Figure 5.1.

It clearly can be seen that different loss functions behave differently in such situations by covering different parts of the mis-classification spectrum. AdaBoost, which uses the exponential loss, has the most aggressive penalty for a mis-classification. This justifies why AdaBoost dramatically increases the weight of mis-classified samples. Going further,

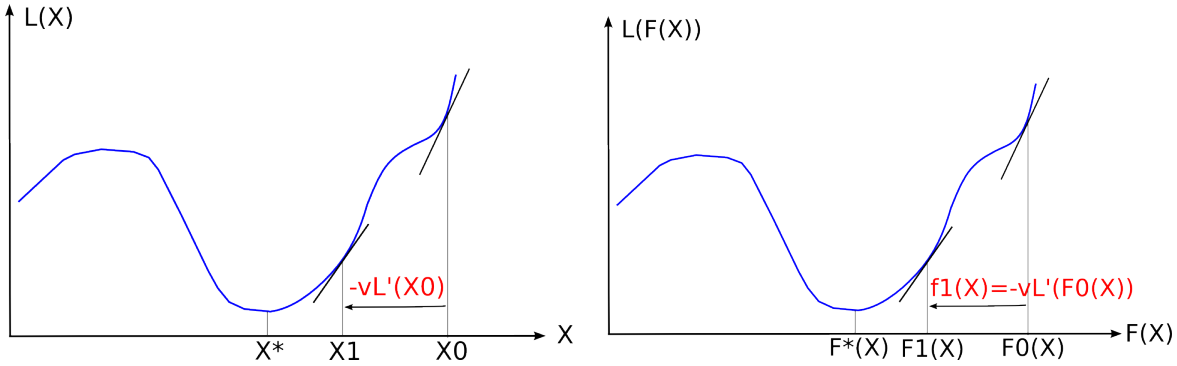
one can see that Logit and Hinge losses are less aggressive and their penalty increases linearly on the left side of the figure. In contrast, DoomII and Savage follow totally different strategies. First, their loss-functions are non-convex. Second, they almost give up on putting pressure over the classifier when there is a severe mis-classification (*i.e.*, $F(x)$ is large). Notably, DoomII gives up much earlier but maintains a higher overall penalty compared to the Savage loss.

5.2.2 On-line GradientBoost

To allow a comparison of different loss functions, we propose an on-line formulation of the GradientBoost [Friedman, 2001] for feature selection, which we call *On-line GradientBoost*. GradientBoost performs stage-wise functional gradient descent over a given loss function [Mason et al., 1999, Friedman et al., 2000, Friedman, 2001]. For an illustration see also Figure 5.2. More specifically, for a loss $\ell(\cdot)$, we want to find a set of base functions or weak learners $\{f_1(x), \cdots, f_M(x)\}$ and their corresponding boosting model

$$F(\mathbf{x}) = \sum_{m=1}^{M} f_m(\mathbf{x}) \tag{5.6}$$

which minimizes this loss.



(a) Gradient Descent (b) Functional Gradient Descent

Figure 5.2: Comparison of common gradient descent (a) and functional gradient descent (b).

Given a loss function $\ell(\cdot)$ and a training dataset, $\mathcal{X} = \{(x_1, y_1), \cdots, (x_N, y_N)\}, x_n \in \mathbb{R}^D, y_n \in \{-1, +1\}$, the empirical loss is

$$\mathcal{L}(F(\mathbf{x})) = \sum_{n=1}^{N} \ell(y_n F(\mathbf{x}_n)). \tag{5.7}$$

According to the gradient descent principle in optimization, at stage t of Gradient-Boost, we are looking for a base function which has the maximum correlation with negative direction of the loss function. This can be written as

$$f_t(\mathbf{x}) = \arg\max_{f(x)} \; -\nabla\mathcal{L}^T f(\mathbf{x}), \tag{5.8}$$

where $\nabla\mathcal{L}$ is the gradient vector of the loss at $F_{t-1}(x) = \sum_{m=1}^{t-1} f_m(x)$. This can be simplified to

$$f_t(\mathbf{x}) = \arg\max_{f(x)} \; -\sum_{n=1}^{N} y_n \ell'(y_n F_{t-1}(\mathbf{x}_n)) f(\mathbf{x}_n). \tag{5.9}$$

where $\ell'(\cdot)$ shows the derivatives of the loss with respect to F_{t-1}.

This can also be written as the maximum projection of the weak learner on to the negative gradient via the inner product:

$$f_t(\mathbf{x}) = \arg\max_{f\in\mathcal{F}} \langle -\nabla\mathcal{L}^T(\mathcal{X}, f(\mathcal{X})\rangle, \tag{5.10}$$

where $\mathcal{F} = \{f_1, \ldots, f_m\}$ is the set of weak learners. See also Figure 5.3 for an illustration of the resulting boosting principle.

The sample weights are calculated as $w_n = -\ell'(y_n F_{t-1}(x_n))$. Thus, the optimization is equivalent to maximizing the weighted classification accuracy

$$f_t(x) = \arg\max_{f(x)} \sum_{n=1}^{N} w_n y_n f(x_n). \tag{5.11}$$

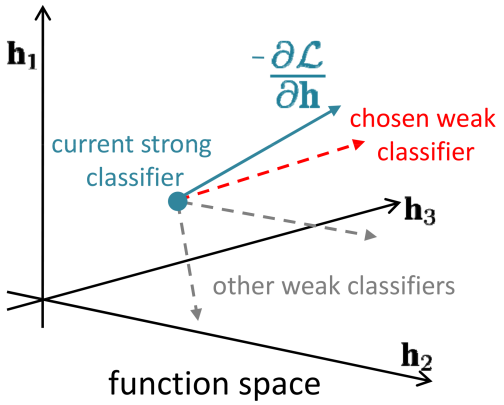


Figure 5.3: Illustration of Gradient Boosting.

On-line Learning

Based on these derivations, we propose an on-line version of GradientBoost, which we show in Algorithm 5.2. As in [Grabner and Bischof, 2006] we keep a fixed set of weak learners and perform boosting on the selectors. The m^{th} selector maintains a set of K weak learners $\mathcal{S}_m = \{f_m^1(x), \cdots, f_m^K(x)\}$ and at each stage it selects the best performing weak learners. The optimization step Equation (5.11) is then performed iteratively by propagating the samples through the selectors and updating the weight estimate λ_m according to the negative derivative of the loss function. Thus, the algorithm is independent of the used loss function.

Algorithm 5.2 On-line GradientBoost

Require: A training sample: (x_n, y_n).
Require: A differentiable loss function $\ell(\cdot)$.
Require: Number of selectors M.
Require: Number of weak learners per selector K.

Set $F_0(x_n) = 0$.
Set the initial weight $w_n = -\ell'(0)$.
for $m = 1$ to M **do**
    for $k = 1$ to K **do**
        Train k^{th} weak learner $f_m^k(x)$ with sample (x_n, y_n) and weight w_n.
        //Compute the error
        $e_m^k \leftarrow e_m^k + w_n \mathbb{I}(\text{sign}(f_m^k(x_n)) \neq y_n)$.
    end for
    Find the best weak learner with the least total weighted error: $j = \arg\min_k e_m^k$.
    Set $f_m(x_n) = f_m^j(x_n)$.
    Set $F_m(x_n) = F_{m-1}(x_n) + f_m(x_n)$.
    Set the weight $w_n = -\ell'(y_n F_m(x_n))$.
end for

Output the final model: $F(x)$

In Figure 5.4 we plot the functions for the weight updates for different popular loss functions. As can be seen, the exponential loss has an unbounded weight update function, while all other loss functions are bounded between $[0, 1]$. Most importantly, Logit and Hinge weight update functions saturate at 1, as the margin decreases, while the weights of Doom and Savage fades out. This also illustrates the success of SavageBoost on noisy samples; *i.e.*, in contrast to AdaBoost if a sample is misclassified with high accuracy it is not incorporated into the learning with exponentially growing weight but it is considered

as outlier and the weight is thus decreased. LogitBoost can be considered as lying between these two extremes; *i.e.*, if a sample is misclassified with high accuracy the logit-loss neither increases the weight nor decreases the weight but keeps it constant to one.

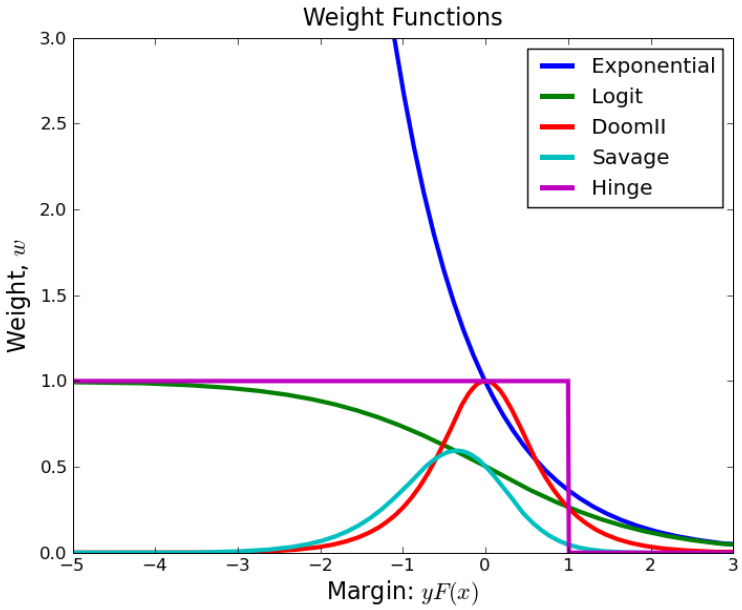


Figure 5.4: Weight update functions for different loss functions.

5.2.3 Competitive Study

In this section, we will conduct a competitive study with the proposed on-line GradientBoost on machine learning data in order to study the influence of the different loss functions. Therefore, we chose 8 benchmark datasets from UCI and LIBSVM repositories, which are shown in Table 6.1. We compare the performance of on-line *AdaBoost* and *GradientBoost* by using exponential, Logit, DoomII, and Savage losses. Note, when using exponential loss, we get the on-line formulation of RealBoost of Friedman *et al.* [Friedman et al., 2000]. For these experiments, we randomly introduce label noise into the training set and train the on-line classifiers for 5 epochs. We repeat all these experiments for 3 times and report the average test errors.

We also repeat these experiments for two different kinds of weak learners, *i.e.*, (i) decision stumps which assume the feature responses being Gaussian distributed, where the means μ^+, μ^- and the standard deviations σ^+, σ^- are estimated with the help of a Kalman filter [Grabner and Bischof, 2006], and (ii) fixed-binned on-line histograms. Some variants of on-line GradientBoost, *e.g.*, RealBoost, need confidence-rated weak

predictions which can be easily calculated using on-line histograms as weak learners. Following [Friedman et al., 2000] we use the probabilistic output of the classifier in form of

$$f(x) = \frac{1}{2} \log \frac{p_+(x)}{1 - p_+(x)}, \tag{5.12}$$

where $p_+(x)$ is the probability of a sample to be positive. Histograms can also inherently describe multi-modal distributions, which as we show in the experiments is also helpful for algorithms which in principle do not require confidence-rated predictions.

In total we conducted 192 experiments per classifier, which hopefully shows clearly which methods perform best in presence of noise. For multi-class datasets, we used an one-vs-all strategy and incorporated the ratio between positive and negative samples in the initial weight of samples.

Dataset	# Train	# Test	# Class	# Feat.
DNA	1400	1186	3	180
Letter	15000	5000	26	16
Magic	9510	9510	2	10
Pendigits	7494	3498	10	16
SatImage	3104	2000	6	36
Shuttle	30450	14500	7	9
Splice	1000	2175	2	60
USPS	7291	2007	10	256

Table 5.2: Datasets used in machine learning experiments.

Figures 5.5(a) and 5.5(b) show the test error with respect to the amount of label noise in the training set for each classifier and each dataset by using stumps and histograms, respectively. The main outcome of these experiments is that on average On-line GradientBoost using Logit or Savage losses performs best. Even though using the DoomII loss function provides excellent results outperforming the others for some datasets, the results obtained by using Logit and Savage loss-functions are consistently among the best.

Additionally, as it can be seen, the Real loss-function does not perform very well when using stumps, but is competitive when the weak learner is switched to histograms. The reason for this behavior could be attributed to the fact that stumps are not able to return probabilistic outputs which are required by on-line GradientBoost losses. By referring to Figure 5.4, we can see that for the Real loos-function does not provide a bounded weight update function, and therefore, without a probabilistic estimates, the sample weights grow unbounded. Additionally, by comparing these two figures it becomes clear that, in gen-

eral, histograms are better than stumps; especially, for AdaBoost and RealBoost significant improvements can be achieved.

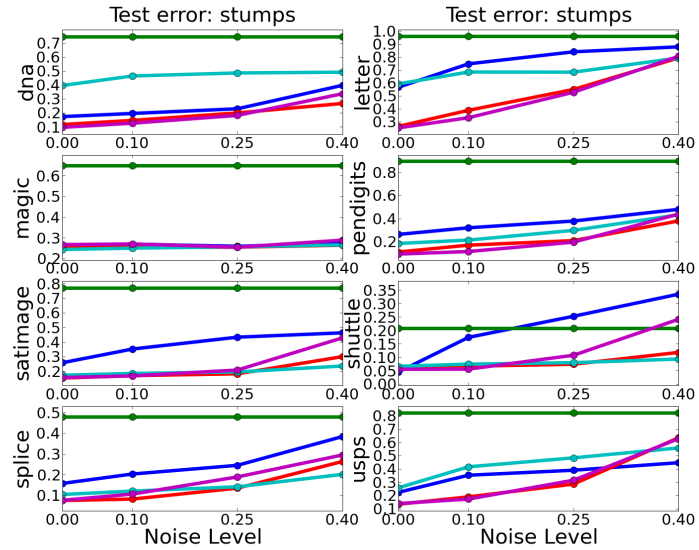
Figure 5.6 shows the number of wins for each classifier over different noise levels over all datasets and weak learners. As can be seen, SavageBoost wins more than all other methods, but LogitBoost gains more and more wins if the noise level is increased, while SavageBoost has no win when the noise level is at its maximum.

5.2.4 Autonomous Training of Scene-Specific Person Detectors

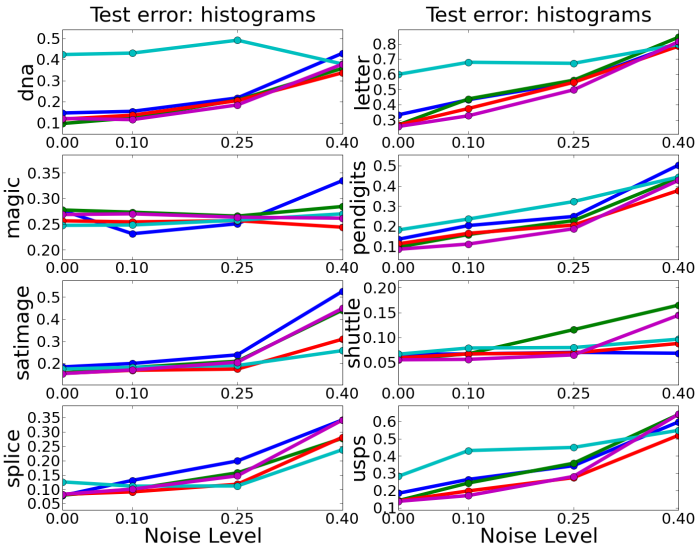
Finally, we analyze on-line boosting when learning object detectors based on co-training [Blum and Mitchell, 1998]. The idea is to train a scene-specific highly accurate and compact classifier with a limited amount of labeled data. The usefulness of such an approach has been demonstrated by Levin *et al.* [Levin et al., 2003] using off-line boosting and by Javed *et al.* [Javed et al., 2005] using the on-line version proposed by Oza. As we reviewed in Section 3.5, for co-training two initial classifiers h_1 and h_2 are trained on some labeled data $\mathcal{D}^L$. Then, these classifiers update each other using an unlabeled data set $\mathcal{D}^U$ based on their confidence-rated predictions. Abney [Abney, 2007] showed that co-training classifiers aim to minimize the error on the labeled samples while increasing the agreement on the unlabeled data. Thus, the unlabeled data helps to improve the margin of the classifiers and to decrease the generalization error. However, in this way also wrong updates may be performed, which arises the need for robust methods. Co-training is thus a perfect application to test the practical impact of robust learners.

Therefore, we compared on-line GradientBoost to a state-of-the-art on-line AdaBoost [Grabner and Bischof, 2006] classifier for training scene-specific person detectors. For GradientBoost, we chose the logistic loss, since the theoretical considerations as well as experimental results on the machine learning data show that this loss can be considered a suitable trade-off between accuracy and robustness. Note that we skipped the results for Savage-loss since it performs similar to the logit-loss. We also use histograms with 32 bins as weak learners. For both, AdaBoost and GradientBoost, the same implementation framework with the same features was used. Since the main purpose of this experiment is the comparison of the two on-line algorithms for the detection task, we only use simple Haar-features [Viola and Jones, 2002a], and generate the different feature "views" via randomly subsampling from a large overcomplete feature set. For both on-line boosting methods we used 50 selectors each of them having 150 features. Note that [Levin et al., 2003, Javed et al., 2005] trained two classifiers using two different representations, one for gray-value images and the other from background subtracted images.

In particular, we trained the initial classifiers using only 25 labeled positives samples, which are then updated by on-line co-training. To demonstrate the learning progress,

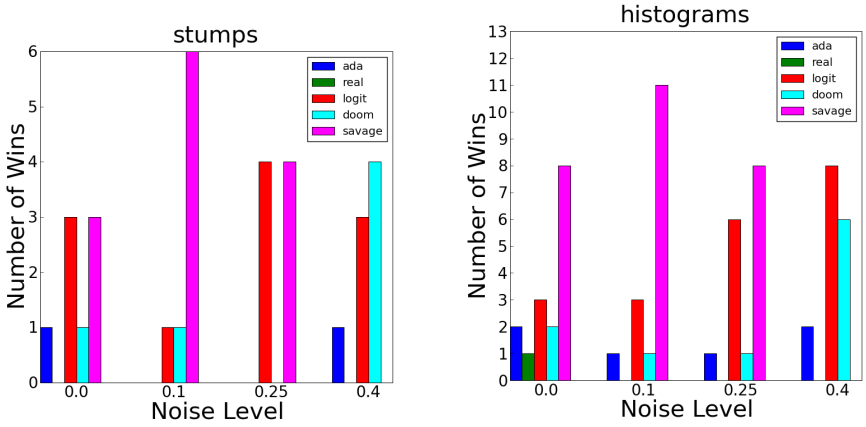


(a) Stumps



(b) Histograms

Figure 5.5: Results of the machine learning experiments when (a) stumps and (b) histograms are used as weak learners: test error is shown with respect to the label noise level. Classifiers: AdaBoost (blue), Real (green), Logit (red), DoomII (cyan), Savage (magenta).

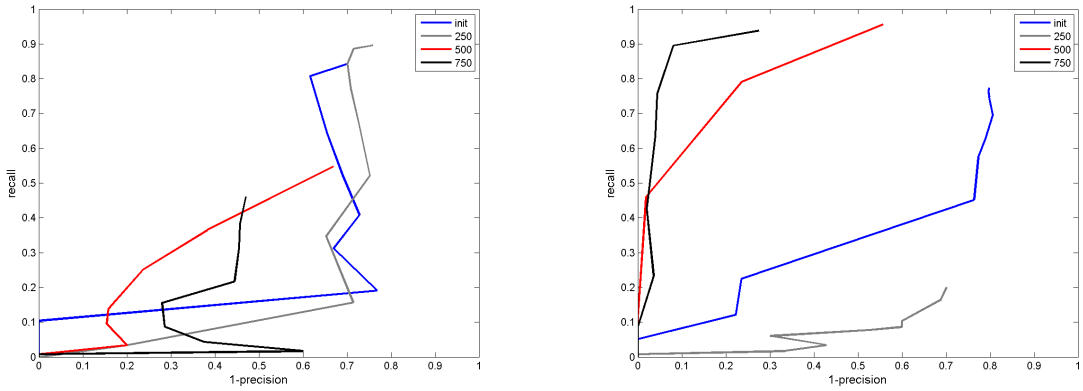


(a) Stumps (b) Histograms

Figure 5.6: Different number of wins for the machine learning experiments when (a) stumps and (b) histograms are used as weak learners: test error is shown with respect to the label noise level. Classifiers: AdaBoost (blue), Real (green), Logit (red), DoomII (cyan), Savage (magenta).

after a pre-defined number of processed training frames t we saved a classifier (*i.e.*, $t = 0$, $t = 250$, $t = 500$, and $t = 750$), which was then evaluated on an independent test sequence (*i.e.*, the current classifier was evaluated but no updates were performed!). To analyze the detection results, we compare $precision$-$recall$-curves (RPC) obtained from a a 50% overlap criterion. The corresponding curves for AdaBoost and GradientBoost are illustrated in Figure 5.7(a) and Figure 5.7(b), respectively. It clearly can be seen that using the non-robust learner the system fails completely. In fact, the wrong updates cannot be handled and the classifier is degraded. In contrast, even starting from a similar initial level, using GradientBoost, finally, a competitive classifier can be obtained.

Discussion We have seen that boosting is highly susceptible to noisy data. Nevertheless, the on-line version is often used in self-learning applications where an adaptive learner is needed but noise is an inherent problem. For the off-line case in principle there exist two approaches to remedy the sensitivity to noise; *i.e.*, heuristics and the design of smarter loss-functions. As heuristics often perform well in case of noise they have the problem that they usually cannot take pace with AdaBoost in case of "clean" training samples whereas smarter loss function design has shown to fulfill this requirement. Therefore, we introduced on-line GradientBoost that provides a flexible way of using any robust loss and demonstrated in the experiments that in practice this can lead to much better results. Thus, in the following we will incorporate the gained insights of this section and propose



(a) AdaBoost (b) GradientBoost

Figure 5.7: *Co-training* using (a) the non-robust on-line AdaBoost algorithm and (b) the robust On-line GradientBoost algorithm.

a more robust semi-supervised boosting method based on the logistic loss.

5.3 On-line SERBoost

Based on the above introduced on-line GradientBoost, we now can also develop an on-line semi-supervised boosting algorithm that performs functional gradient descent and is thus able to incorporate arbitrary loss functions. Since for the off-line case, SERBoost [Saffari et al., 2008] is a popular semi-supervised boosting algorithm that is based on GradientBoost (see also Section 3.7.1) it is also a promising candidate for our on-line semi-supervised method. However, SERBoost as proposed in [Saffari et al., 2008] uses the exponential loss. Hence, in the following we first derive SERBoost for the logistic loss and then present the on-line version.

In detail, we use the following log-likelihood for the loss over labeled samples

$$\begin{aligned}\mathcal{L}_l(\mathcal{X}^L) &= \sum_{(\mathbf{x},y)\in\mathcal{X}_l} \log\left(1+e^{-2yF(\mathbf{x})}\right) \\ &= \sum_{(\mathbf{x},y)\in\mathcal{X}^L} \log\left(e^{-yF(\mathbf{x})}(e^{yF(\mathbf{x})}+e^{-yF(\mathbf{x})})\right) \\ &= \sum_{(\mathbf{x},y)\in\mathcal{X}^L} -yF(\mathbf{x}) + \log\left(e^{F(\mathbf{x})}+e^{-F(\mathbf{x})}\right).\end{aligned} \tag{5.13}$$

SERBoost performs SSL based on given priors which encode designed assumptions over the unlabeled data. [Saffari et al., 2008] use the Kullback-Leibler (KL) divergence

between the prior probability and the optimized model in order to define the loss function for the unlabeled data. The optimization reduces to minimizing the cross entropy

$$\begin{aligned} H(P_p, P) &= -\sum_{z\in\{-1,1\}} P_p(y=z|\mathbf{x}) \log P(y=z|\mathbf{x}) \\ &= -\underbrace{(2P_p(y=1|\mathbf{x})-1)}_{y_p(\mathbf{x})} F(\mathbf{x}) + \log\left(e^{F(\mathbf{x})} + e^{-F(\mathbf{x})}\right). \end{aligned} \tag{5.14}$$

Thus, the loss over unlabeled samples results in

$$\mathcal{L}_u(\mathcal{X}^U) = \sum_{\mathbf{x}\in\mathcal{X}^U} H(P_p, \hat{P}) = \sum_{\mathbf{x}\in\mathcal{X}^U} -y_p(\mathbf{x})F(\mathbf{x}) + \log\left(e^{F(\mathbf{x})} + e^{-F(\mathbf{x})}\right). \tag{5.15}$$

Here, $y_p(\mathbf{x}) = 2P_p(y=1|\mathbf{x}) - 1 \in [-1,1]$ is the prior soft label. For the prior probability $P_p(y=1|\mathbf{x})$ any available prior information can be used. Since the formulation of Equation (5.15) is the logit loss of Eq (5.13) in case the prior soft label $y_p(\mathbf{x})$ is casted to a hard label $\in \{-1,1\}$, we do not further pass it to an exponential function as in [Saffari et al., 2008]. As a result, both labeled and unlabeled loss are based on a coherent logistic function.

5.3.1 On-line learning

In order to minimize the loss we use the on-line GradientBoost as introduced above. As Grabner *et al.*, we keep a fixed set of weak learners and perform boosting over the selectors. The semi-supervised loss is iteratively optimized by propagating the samples through the selectors and updating the weight estimate λ_m according to the negative derivative of the loss. Hence, we need to compute the *negative* gradients a_{ij} with respect to the current classifier of the logit loss function:

$$a_{\mathbf{x}}(z) = \frac{\partial\, zF(\mathbf{x}) - \log\left(e^{F(\mathbf{x})} + e^{-F(\mathbf{x})}\right)}{\partial F(\mathbf{x})} = z - \tanh(F(\mathbf{x})), \tag{5.16}$$

where z is either the hard label y for labeled samples, or the soft prior label $y_p(\mathbf{x})$ for unlabeled samples. Therefore, the update rule for unlabeled samples can be written as

$$\begin{aligned} \forall\, \mathbf{x} \in \mathcal{X}^U : w_{\mathbf{x}} &= |y_p(\mathbf{x}) - \tanh(F(\mathbf{x}))| \\ \hat{y}_{\mathbf{x}} &= \operatorname{sign}\left(y_p(\mathbf{x}) - \tanh(F(\mathbf{x}))\right) \end{aligned}$$

We depict a summary of the update rules for unlabeled samples depending on popular loss functions in Table 5.3 and show the entire algorithm in Algorithm 5.3.

Dataset	KL-Exponential [Saffari et al., 2008]	KL
$w_{\mathbf{x}}^U \leftarrow$	$\lvert y_p(\mathbf{x})\cosh(F(x)) - \sinh(F(x))\rvert e^{-y_p F(x)}$	$\lvert y_p(\mathbf{x}) - \tanh(F(x))\rvert$
$\hat{y}_{\mathbf{x}} \leftarrow$	$\text{sign}(y_p(\mathbf{x})\cosh(F(x)) - \sinh(F(x)))$	$\text{sign}(y_p(\mathbf{x}) - \tanh(F(x)))$

Table 5.3: Comparison of update rules depending on different loss functions; *i.e.*, exponential and logit loss.

5.4 Machine Learning Experiments

In this set of experiments, we evaluated on-line SemiBoost and on-line SERBoost on standard semi-supervised benchmark data sets taken from [Chapelle et al., 2006] [1]. The data consists of both artificial and real-life data. Furthermore, on some data sets the cluster assumption holds while on some the manifold assumption holds. A summary of the data sets is presented in Table 5.4. We compared our methods to supervised off-line variants of nearest neighbor (1-NN), SVM and AdaBoost. We used LaRank SVM [Bordes et al., 2007] on-line GradientBoost with logistic loss (OLogitBoost) [Leistner et al., 2009a] as well as standard on-line boosting (OAdaBoost) [Grabner and Bischof, 2006] with LaRank as weak learners. For semi-supervised comparison we took the off-line versions of SERBoost [Saffari et al., 2008], ManifoldBoost [Loeff et al., 2008] and TSVM [Joachims, 1999]. For both on-line and off-line SERBoost we used k-means clustering as prior. For SemiBoost we used the Euclidean distance in order to calculate S_{ij}, with σ set using 5-fold cross validation. For all boosting methods we used 50 weak learners. For gradient-based methods we set the shrinkage factor to $\nu = 0.1$. The importance λ of the unlabeled data was set to 0.1. We present the results in Table 5.5.

As can be seen, both on-line SemiBoost and on-line SERBoost are competitive SSL methods and both are able to match the results of their off-line counterparts. As expected, SemiBoost has slight advantages on manifold-like data sets while SERBoost performs very well on cluster-based data sets. Interestingly, OSemiBoost is able to match the performance of OSERBoost on $g241c$, a manifold-based set, with $l = 10$ and OSERBoost is able to outperform OSemiBoost on $Digit1$, a cluster-based set, with $l = 10$.

5.5 Summary and Conclusion

In this chapter, we introduced a novel on-line semi-supervised boosting algorithm based on SemiBoost. The algorithm is thus called on-line SemiBoost. We further illustrated that one of the major drawbacks of current boosting algorithms, both off-line and on-line,

[1] http://www.kyb.tuebingen.mpg.de/ssl-book

Algorithm 5.3 On-line SERBoost with logistic loss

Require: A training sample: $(x_n, y_n) \in \mathcal{X}_L$ or $(x_n) \in \mathcal{X}_U$.
Require: Prior knowledge: $\forall(\mathbf{x}_i, y_i) \in \mathcal{X}_U, y : P_p(y|\mathbf{x})$
Require: Number of selectors M.
Require: Number of weak learners per selector K.

Set $F_0(x_n) = 0$.
if $(x_n, y_n) \in \mathcal{X}_L$ **then**
    Set the initial weight $w_n = \frac{1}{2}$
else
    Set the initial weight $w_n^U = |y_p(\mathbf{x}_i) - \tanh(0)| = |y_p(\mathbf{x}_i)|$.
    Set the initial label $\hat{y}_n = \text{sign}\left(y_p(\mathbf{x}_n)\right)$.
end if
for $m = 1$ to M **do**
    if $(x_n, y_n) \in \mathcal{X}_L$ **then**
        Set the weight $w_n = \frac{1}{1+e^{y_n F_m(x_n))}}$.
    else
        // Estimate unlabeled weights and pseudo labels
        Set the weight $w_n^P = |y_p(\mathbf{x}_i) - \tanh(y_n F_m(x_n))|$.
        Set the label $\hat{y}_n = \text{sign}\left(y_p(\mathbf{x}_i) - \tanh(y_n F_m(x_n)\right)$.
    end if
    for $k = 1$ to K **do**
        Train k^{th} weak learner $f_m^k(x)$ with sample (x_n, y_n) and weight w_n.
        //Compute the error
        $e_m^k \leftarrow e_m^k + w_n \mathbb{I}(\text{sign}(f_m^k(x_n)) \neq y_n)$.
    end for
    Find the best weak learner with the least total weighted error: $j = \arg\min_k e_m^k$.
    Set $f_m(x_n) = f_m^j(x_n)$.
    Set $F_m(x_n) = F_{m-1}(x_n) + f_m(x_n)$.
    Set the weight $w_n = \frac{1}{1+e^{y_n F_m(x_n))}}$.
end for

Output the final model: $F(x)$

is that they are highly susceptible to class label noise and that this is mostly due to the usage of non-robust loss functions. Based on this insight we proposed an on-line version of GradientBoost, which performs boosting as functional gradient descent and allows for easy incorporation of robust loss functions. Based on on-line GradientBoost we further

Dataset	# Samples	Dimension	# Class	Comment
BCI	400	117	2	manifold
Digit1	1500	241	2	manifold
g241d	1500	241	2	cluster
g241c	1500	241	2	cluster

Table 5.4: Data sets for the machine learning experiments.

Method	Manifold-like				Cluster-like			
	l=10		l=100		l=10		l=100	
	BCI	Digit1	BCI	Digit1	g241c	g241d	g241c	g241d
Supervised Methods								
SVM	49.85	30.6	34.31	5.53	47.32	46.66	23.11	24.64
1-NN	49	13.65	48.67	3.89	47.88	46.72	43.93	42.45
OAdaBoost	46.9	18.9	24.6	8.2	36.1	42.7	22.8	26.1
LaRank	46.9	18.9	25.6	9.5	36.1	42.7	23.4	26.1
OLogitBoost	46.5	18.9	**24.5**	8.3	35.9	42.7	22.7	26.3
Semi-Supervised Methods								
TSVM	49.15	17.77	33.25	6.15	24.71	50.08	18.46	22.42
ManifoldBoost	47.12	19.42	32.17	4.29	42.17	42.8	22.87	25
SERBoost	46.9	**15**	49	13.1	13.1	**4.4**	**13.2**	**4.4**
SemiBoost-RF	49.77	10.57	47.12	2.56	48.41	41.26	47.19	39.14
OSemiBoost	**37.1**	19.1	26.3	**5.1**	**12.9**	38.6	16.5	**4.4**
OSERBoost	42.5	16.7	29	14.9	**12.9**	**4.4**	14.2	**4.4**

Table 5.5: Classification error on semi-supervised learning benchmark data sets for 10 and 100 labeled samples, respectively. The upper half evaluates super-vised methods, the lower part semi-supervised approaches. For both supervised and semi-supervised approches, we depict results achieved both off-line and on-line. The best performing method is marked bold.

proposed on-line SERBoost using logistic loss functions and thus allows the usage of more robust loss functions for on-line semi-supervised boosting. In Chapter 9, we will depict detailed evaluations of the methods for the task of object tracking.

Chapter 6

Semi-Supervised Random Forests

In Chapter 3, we have seen that there exist various semi-supervised learning approaches, many of them based on two of the most successful learning algorithms in the machine learning field, *i.e.*, support vector machines [Schoelkopf and Smola, 2002] and boosting [Freund and Schapire, 1999]. Yet, as we discussed in Chapter 2, for supervised learning, recently random forests [Breiman, 2001] emerged as an interesting alternative to SVMs and boosting. What makes random forests interesting, especially to computer vision, is their speed during both training and evaluation along with the fact that they are perfectly suitable for multi-core architectures. In addition, they are inherently multi-class and are more noise tolerant than other state-of-the-art methods, especially compared to boosting.

However, random forests suffer from the same disadvantages as other popular discriminative learning methods: they need a huge amount of labeled data in order to achieve good performance. Even worse, due to their bagging and tree nature it has been shown [Caruana et al., 2008] that on many data sets they even demand more training data than other methods in order to develop their full potential and to outperform competitive approaches. This rises the suggestion that RFs would presumably benefit even more from large amounts of unlabeled data than competitive methods.

In this chapter, we propose a novel semi-supervised learning algorithm for RFs allowing the algorithm to make use of both labeled and unlabeled training data. In fact, the beneficial characteristics of random forests as stated above make them also interesting candidates for semi-supervised learning. Especially their ability to handle multi-class tasks makes RFs very attractive for SSL problems because most previous semi-supervised methods only focus on binary problems. Multi-class problems are often decomposed to a set of binary tasks with 1-vs-all or 1-vs-1 strategies. Considering the fact that most of the state-of-the-art SSL methods have high computational complexity, such a strategy can become a problem when dealing with a large number of samples and classes.

6.1 Semi-Supervised Learning with Random Forests

As discussed in Chapter 3, there exist various regularization approaches to semi-supervised learning, where the large-margin-assumptions and manifold assumptions are the most popular ones. Since we target applications with a large amount of data and manifold regularization leads to algorithms that are at least quadratic, *i.e.* $\mathcal{O}(n^2)$, in terms of the number of samples, in this work, we chose to use a maximum margin approach. In particular, we propose to exploit the additional unlabeled data in order to maximize the margin over the entire random forest. As RFs are multi-class classifiers, in the following we will define the margin maximizing properties of decision trees for multi-class problems and subsequently define the margin over the unlabeled samples.

Based on this max-margin definition we incorporate the hidden labels of unlabeled samples as additional optimization variables. Our overall optimization problem will then become non convex. In order to solve this problem, we use a deterministic annealing-style optimization strategy which preserves the diversity among the trees and, due to its random nature, both fits perfectly to the nature of randomized trees and does not slow-down the training speed significantly.

6.1.1 Margin for Multi-Class Classification

Recently, Zou *et al.* [Zou et al., 2008] extended the concept of Fisher-consistent loss functions [Lin, 2004] from binary classification to the domain of multi-class problems. This concept provides an understanding about the success of margin-based loss functions and their statistical characteristics.

Let $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \cdots, g_K(\mathbf{x})]^T$ be a multi-valued function; *i.e.*, in our case the prediction function of the random forest. $\mathbf{g}(\mathbf{x})$ is called a margin vector, if

$$\forall \mathbf{x} : \sum_{i=1}^{K} g_i(\mathbf{x}) = 0. \tag{6.1}$$

In such a case, one can define the margin for the i^{th} class as $g_i(\mathbf{x})$ and the true margin as $g_y(\mathbf{x})$. For a Fisher consistent loss function, this quantity is naturally linked to the optimal Bayes' decision rule [Zou et al., 2008]; which means that all learners that are based on Fisher consistent loss functions and are able to minimize them inherently approximate Bayes' decision rule.

We define a loss function $\ell(g_y(\mathbf{x}))$ to be a margin maximizing loss if $\ell(g_y(\mathbf{x}))$ is differentiable and for the first derivative $\ell'(g_y(\mathbf{x})) \leq 0$ for all values of g_y. Therefore, an optimization based on this kind of loss functions will lead to maximizing the true margin. In this respect, the exponential loss of boosting, the hinge loss of SVMs, and the negative log-likelihood loss of statistical models are all margin maximizing loss functions.

For decision trees, local tests at each node are selected based on a score which measures the purity of the node. As we have seen in Chapter 2, usual choices are the entropy ($\mathcal{L}(\mathcal{R}_j) = -\sum_{i=1}^{K} p_i^j \log(p_i^j)$) or the Gini index ($\mathcal{L}(\mathcal{R}_j) = \sum_{i=1}^{K} p_i^j(1-p_i^j)$), where p_i^j is the label density of class i in node j. In the following Theorem, we relate such scores to multi-class margin maximizing loss functions and show that one can pick any margin maximizing loss function and derive a local score measurement.

Theorem 6.1.1. *Given a margin maximizing loss function $\ell(g_y(\mathbf{x}))$, the local score for a decision node $\mathcal{R}_j$ is defined as $\mathcal{L}(\mathcal{R}_j) = \sum_{i=1}^{K} p_i^j \ell(p_i^j - \frac{1}{K})$.*

Proof: We can write the empirical loss at this node as

$$\mathcal{L}(\mathcal{R}_j) = \frac{1}{|\mathcal{R}_j|} \sum_{(\mathbf{x},y)\in\mathcal{R}_j} \ell(g_y(\mathbf{x})). \tag{6.2}$$

Defining the margin vector as $\mathbf{g}^j(\mathbf{x}) = [p_1^j - \frac{1}{K}, \cdots, p_K^j - \frac{1}{K}]^T$, we can develop the empirical loss as

$$\begin{aligned}
\mathcal{L}(\mathcal{R}_j) =& \frac{1}{|\mathcal{R}_j|} \sum_{(\mathbf{x},y)\in\mathcal{R}_j} \sum_{i=1}^{K} \mathbb{I}(y=i)\ell(p_i^j - \frac{1}{K}) \\
=& \frac{1}{|\mathcal{R}_j|} \sum_{i=1}^{K} \ell(p_i^j - \frac{1}{K}) \sum_{(\mathbf{x},y)\in\mathcal{R}_j} \mathbb{I}(y=i) \\
=& \sum_{i=1}^{K} p_i^j \ell(p_i^j - \frac{1}{K}).
\end{aligned}$$

□

Using the results of Theorem 6.1.1, we can see that the entropy score minimizes the calibrated negative log-likelihood while the Gini index is related to the hinge loss function. Thus, we can conclude that traditional decision trees greedily optimize multi-class maximum margin criteria.

6.1.2 Margin for the Unlabeled Data

Now that we have a better understanding with respect to the maximum margin behavior of the decision trees, the extension of this concept to unlabeled data is straight-forward. In the absence of a label, there is no known true margin, therefore, we define the margin as:

$$m_u(\mathbf{x}_u) = \max_{i\in\mathcal{Y}} g_i(\mathbf{x}_u). \tag{6.3}$$

Note that the predicted label for an unlabeled sample is $C(\mathbf{x}) = \arg\max_{i \in \mathcal{Y}} g_i(\mathbf{x}_u)$. Hence, this is equivalent to the margin of the labeled samples, given in Eq (2.14), but only using the predicted label.

6.1.3 Learning

Similar to the traditional regularization-based semi-supervised learning methods, we also regularize the loss for the labeled samples with a loss over the unlabeled samples. Based on the definition of the margin for unlabeled samples, we use the same loss function used to grow the trees in a forest also to be the loss for the unlabeled samples. We can write the overall loss as

$$\begin{aligned}\mathcal{L}(\mathbf{g}) =& \frac{1}{|\mathcal{X}_l|} \sum_{(\mathbf{x},y)\in\mathcal{X}_l} \ell(g_y(\mathbf{x})) + \\ &+ \frac{\alpha}{|\mathcal{X}_u|} \sum_{\mathbf{x}\in\mathcal{X}_u} \ell(m_u(\mathbf{x})),\end{aligned} \tag{6.4}$$

where α defines the contribution rate of the unlabeled samples.

Note that when training only on labeled data, usually convex loss functions are used (standard RF algorithm). Using additional unlabeled data makes the loss non-convex because also the hidden labels $\hat{y}_u$ of the unlabeled samples have to be optimized. In fact, due to the integer values of $\hat{y}_u$ the problem is a type of integer programming which is $\mathcal{NP}$-complete [Boyd and Vandenberghe, 2004]. Thus, we need a (global) optimization method over the forest that is highly resistant to local minima.

6.1.4 Optimization

In this work, we formulate the optimization process in a deterministic annealing (DA) framework. DA is a homotopy method where an eventually difficult combinatorial optimization problem is rewritten in an easier form and then gradually deformed to its original version [Rose, 1998]. It emerged from simulated annealing optimization techniques [Kirkpatrick et al., 1983] which is based on a sequence of random moves and the cost to accept a move depends on the cost of changing the current state. The frequency of the random moves is reduced from high values to zero depending on an *annealing schedule* which successively reduces a so called temperature parameter T, *i.e.*, $T_0 > T_1 > \ldots > T_\infty = 0$. Theoretically, simulated annealing can find a global minimum, however, using unrealistic annealing schedules. Deterministic annealing, as the name suggests, brings determinacy into to annealing process by replacing stochastic simulations by the use of expectation. In particular, one tries to minimize the entropy $\mathcal{H}$ of

the distribution p in form of:

$$p^* = \arg\min_{p \in \mathcal{P}} E_p(\mathcal{F}(y)) - T\mathcal{H}(p), \tag{6.5}$$

where $\mathcal{P}$ is a space of probability distributions and $\mathcal{F}(y)$ is our objective function. The optimization problem is than gradually deformed to its original form using a cooling parameter T. In more detail, in a first step the discrete variables are treated as random variables over which a space of probability distributions is defined. In the second step, the original problem is replaced by a continuous optimization term. Although DA cannot guarantee a global optimal solution, the method has proven to be a both fast and robust optimization technique which is able to avoid poor local minima. Note that Sindhwani *et al.* [Sindhwani et al., 2006] used a similar approach for developing a semi-supervised kernel machine. Using DA, we treat unknown labels of the unlabeled samples as additional optimization variables. Additionally, the random nature of DA allows us to keep high diversity among the trees, which is a necessity (as can be seen in Equation (2.16)) for an improvement of the generalization error of the forest.

Based on that, we propose to optimize a regularized loss function with the usage of deterministic annealing. Note that it is not straight-forward to directly apply those ideas to RFs. Firstly, during the training of RFs, the only place where a loss function is used is when selecting a test for a decision node. Since the pool of tests are mainly generated randomly, the overall quality of the chosen decision test is not an important factor (*i.e.*, we are not looking for the best possible decision test, but for a test that performs sufficiently well). Hence, the effect of directly implementing a regularization term inside the local decision nodes is very small, if any. Therefore, we circumvent node-level regularization and apply deterministic annealing on the forests level, however, still training each tree independently.

6.1.4.1 Deterministic Annealing

We apply deterministic annealing to iteratively solve Equation (6.4), by introducing a distribution over the predicted labels of unlabeled samples, $\hat{\mathbf{p}}$, and enforcing a controlled uncertainty into the whole optimization process. We write the new loss function as

$$\begin{aligned}\mathcal{L}_{DA}(\mathbf{g}, \hat{\mathbf{p}}) =& \frac{1}{|\mathcal{X}_l|} \sum_{(\mathbf{x},y)\in\mathcal{X}_l} \ell(g_y(\mathbf{x})) + \\ &+ \frac{\alpha}{|\mathcal{X}_u|} \sum_{\mathbf{x}\in\mathcal{X}_u} \sum_{i=1}^{K} \hat{p}(i|\mathbf{x}) \ell(g_i(\mathbf{x})) + \\ &+ \frac{T}{|\mathcal{X}_u|} \sum_{\mathbf{x}\in\mathcal{X}_u} \sum_{i=1}^{K} H(\hat{\mathbf{p}}),\end{aligned} \tag{6.6}$$

where T is the temperature parameter and $H(\hat{\mathbf{p}}) = -\sum_{i=1}^{K} \hat{p}(i|\mathbf{x}) \log(\hat{p}(i|\mathbf{x}))$ is the entropy over the predicted distribution. Note that when the temperature is high, the dominating term is the entropy which needs to be minimized. Hence, at such stages, the model will maintain a large amount of uncertainty. As the system cools down, by decreasing the temperature $T \mapsto 0$, the optimization process will mainly operate over the original loss function Eq (6.4).

For a given temperature level, the learning problem can be written as

$$(\mathbf{g}^*, \hat{\mathbf{p}}^*) = \arg\min_{\mathbf{g}, \hat{\mathbf{p}}} \mathcal{L}_{DA}(\mathbf{g}, \hat{\mathbf{p}}). \tag{6.7}$$

We split this optimization problem up into a two-step convex optimization problem analog to an alternating coordinate descent approach. At the first step, we fix the distribution $\hat{\mathbf{p}}$ and optimize the learning model. In the second step, we treat the hidden labels of unlabeled samples as random binary variables. These random variables are defined over a space of probability distributions $\mathcal{P}$. We then search distributions $\hat{\mathbf{p}} \in \mathcal{P}$ over our unlabeled data which solve our optimization problem in Equation (8.3). Note again that both individual steps are convex optimization problems.

In detail, for a given distribution over the unlabeled samples, we randomly choose a label according to $\hat{\mathbf{p}}$. We repeat this process independently for every tree in the forest. At this stage, the optimization problem for the n^{th} tree becomes

$$\begin{aligned} \mathbf{g}_n^* = \arg\min_{\mathbf{g}} \frac{1}{|\mathcal{X}_l|} \sum_{(\mathbf{x},y)\in\mathcal{X}_l} \ell(g_y(\mathbf{x})) + \\ + \frac{\alpha}{|\mathcal{X}_u|} \sum_{\mathbf{x}\in\mathcal{X}_u} \ell(g_{\hat{y}_u}(\mathbf{x})), \end{aligned} \tag{6.8}$$

where $\hat{y}_u$ is the randomly chosen label for this sample according to the distribution $\hat{\mathbf{p}}$. Since the margin maximizing loss function is convex, this loss function is also convex.

After we trained the random forest, we enter the second stage where we find the optimal distribution according to

$$\begin{aligned} \hat{\mathbf{p}}^* = &\arg\min_{\hat{\mathbf{p}}} \frac{\alpha}{|\mathcal{X}_u|} \sum_{\mathbf{x}\in\mathcal{X}_u} \sum_{i=1}^{K} \hat{p}(i|\mathbf{x}) \ell(g_i(\mathbf{x})) + \\ &+ \frac{T}{|\mathcal{X}_u|} \sum_{\mathbf{x}\in\mathcal{X}_u} \sum_{i=1}^{K} \hat{p}(i|\mathbf{x}) \log(\hat{p}(i|\mathbf{x})). \end{aligned} \tag{6.9}$$

For each sample, we can take the derivatives w.r.t. each class of the distribution and set them to zero to find the optimal solution. In detail, we define:

$$h_i(\hat{\mathbf{p}}, \mathbf{x}) = \hat{p}(i|\mathbf{x})(\alpha \ell(g_i(\mathbf{x})) + T \log(\hat{p}(i|\mathbf{x}))). \tag{6.10}$$

The derivatives of this function can be written as

$$\frac{dh_i}{d\hat{p}_i} = \alpha\ell(g_i(\mathbf{x})) + T\log(\hat{p}(i|\mathbf{x})) + T. \tag{6.11}$$

By setting the derivatives to zero, we get

$$\hat{p}^*(i|\mathbf{x}) = \exp(-\frac{\alpha\ell(g_i(\mathbf{x})) + T}{T})/Z(\mathbf{x}), \tag{6.12}$$

where $Z(\mathbf{x}) = \sum_{i=1}^{K} \hat{p}^*(i|\mathbf{x})$ is the partition function. Note again that when the temperature is high, the distribution is close to a uniform distribution, while at very low temperatures it simulates a Dirac delta function, which is the hard decision rule of Equation (2.13) over the unlabeled data.

6.1.5 Airbag

As we have discussed in Section 3.2, in semi-supervised learning there is no guarantee that unlabeled data always helps, for instance, if the problem structure is badly matched, if the unlabeled data is corrupted or from a different distribution, *etc.*. A nice aspect of random forests is that we can directly monitor the strength of the ensemble by measuring the OOBE for the entire forest at each iteration (see also Section 2.4.2). By considering Equation (2.16) we can see that the strength of the forest has an inverse relationship with the generalization error (*i.e.*, the stronger the forest, the lower is its error). Furthermore, the OOBE has shown to be a good estimate of the generalization error [Breiman, 1996b]. This means we can use the OOBE to estimate our model parameters. Note that this is similar to performing n-fold cross-validation; however, in random forests the OOBE is inherently provided along the learning without any additional steps or costs.

This means that we can also use the OOBE in order to measure the influence when learning from unlabeled data; *i.e.*, measure if it helps or it does not help. However, one problem that might occur with this approach is that due to [Breiman, 1996b] the OOBE is only a low-bias estimate for the generalization error if we have enough data and this is mostly not the case in SSL. Fortunately, over the iterations of the deterministic annealing process and as we will see in the experiments, even if the OOBE does not perfectly reflect the real test error, the development of the OOBE corresponds to the dynamics of the real error.

Hence, we propose to monitor the dynamics of the OOBE of the ensemble in order to estimate the influence of the unlabeled data. In more detail, let $e_{\mathcal{F}}^m$ be the OOBE of the forest at iteration m. Then we monitor the improvements measured by $e_{\mathcal{F}}^{m-1} - e_{\mathcal{F}}^m$ and if this is not positive after a few trials, we stop the training and discard the latest forest.

Algorithm 6.1 Semi-supervised Random Forests

Require: A set of labeled, $\mathcal{X}_l$, and unlabeled data $\mathcal{X}_u$.
Require: The size of the forest: N.
Require: A starting heat parameter T_0 and a cooling function $c(T, m)$
Train the RF: $\mathcal{F} \leftarrow$ trainRF($\mathcal{X}_l$).
Compute the OOBE: $e^0_{\mathcal{F}} \leftarrow$ oobe($\mathcal{F}, \mathcal{X}_l$).
Set the epoch: $m = 0$.
repeat
    Get the temperature: $T_{m+1} \leftarrow c(T_m, m)$.
    Set $m \leftarrow m + 1$.
    $\forall \mathbf{x}_u \in \mathcal{X}_u, k \in \mathcal{Y}$: Compute $p^*(k|\mathbf{x}_u)$.
    for n from 1 to N **do**
        $\forall \mathbf{x}_u \in \mathcal{X}_u$: Draw a random label, $\hat{y}_u$ from $p^*(\cdot|\mathbf{x}_u)$ distribution.
        Set $\mathcal{X}_n = \mathcal{X}_l \cup \{(\mathbf{x}_u, \hat{y}_u)|\mathbf{x}_u \in \mathcal{X}_u\}$.
        Re-train the tree: $f_n \leftarrow$ trainTree($\mathcal{X}_n$).
    end for
    Set $e^m_{\mathcal{F}} \leftarrow$ oobe($\mathcal{F}, \mathcal{X}_l$).
until Stopping condition
if $e^m_{\mathcal{F}} > e^0_{\mathcal{F}}$ **then**
    Reset the RF: $\mathcal{F} \leftarrow$ trainRF($\mathcal{X}_l$).
end if
Output the forest $\mathcal{F}$.

Discussion As we have seen, in contrast to most other SSL methods, semi-supervised random forests provide a principled way to detect if unlabeled data rather harm the system than help. As a reaction, we can use an *airbag mechanism* in order to stop the semi-supervised learning process and use only the labeled data. We call our method deterministic annealing based Semi-Supervised Random Forests (DAS-RF) and show the overall learning and airbag procedures in Algorithm 6.1.

6.2 Prior Regularization

The semi-supervised random forest method as introduced above, *i.e.*, DAS-RF, is able to successfully perform SSL by exploiting the unlabeled data in order to increase the classification margin and does not demand any prior information. However, as we have seen in previous chapters, there exist powerful methods, *e.g.*, [Mann and Mccallum, 2007, Saffari et al., 2008], which have shown that incorporating prior information or designer-provided

expectations into SSL - if available - can lead to simple, well performing and scalable SSL algorithms. Thereby, the prior information can come from various sources of knowledge and does not necessarily have to be highly accurate; for instance, it can come from only the label priors $P_p(y)$, human labelers, from other classifiers or even maximum entropy [Berger et al., 1996], where in the latter case all classes have the identical conditional probability.

In more detail, assume we have given a prior probability in form of

$$\forall \mathbf{x}, k \in \mathcal{Y} : q(k|\mathbf{x}). \tag{6.13}$$

The goal of the learning algorithm is to produce a model, which is able to match the prior probability over the training samples. This divergence can be measured, *e.g.*, in form of Kullback-Leibler (KL) divergence which measures the expected amount of added uncertainty by using the probability distribution of $p(z)$ instead of the true distribution $q(z)$.

$$D_{KL}(q\|p) = H(q,p) - H(q). \tag{6.14}$$

Note that when two distributions q and p are the same, then $D_{KL}(q\|p) = 0$ since there is no added uncertainty. Additionally, note that the KL divergence is not symmetric in the above form with respect to p and q but can be symmetrized by

$$D_{SKL}(q\|p) = \frac{1}{2}(D_{KL}(q\|p) + D_{KL}(p\|q)). \tag{6.15}$$

Inducing prior knowledge at node-level As it was done in [Saffari et al., 2008] for boosting, it is now also natural to incorporate prior knowledge over the learning of the randomized trees by measuring the KL-divergence between the current model and the prior and penalizing high deviations. As we will see in the following, this can be enforced at the node-level of the trees.

In more detail, recall from Section 2.4.2 that during training of a randomized tree each node selects the best split according to some quality measurement which scores the potential information gain

$$\Delta H = -\frac{|I_l|}{|I_l| + |I_r|} H(I_l) - \frac{|I_r|}{|I_l| + |I_r|} H(I_r), \tag{6.16}$$

where I_l and I_r are the left and right subsets of the training data, respectively, and $H(I)$ is the node score, usually measured using the entropy or the Gini. Furthermore, we assume having prior information available in form of a conditional probability distribution $q(y|\mathbf{x})$. We can now use SKL-divergence in order to enforce the node so that it not only minimizes the impurity of the labeled samples but also favors splits that match the prior by decreasing

the cross entropy over the unlabeled samples. The prior-regularized node score can be rewritten as

$$\Delta H^* = \Delta H + \beta \Delta D_{SKL}(q \| \hat{p}), \tag{6.17}$$

where ΔH corresponds to Equation (6.16) and β is a constant steering the influence of regularization. The symmetrized cross entropy information gain can further be written as

$$\Delta D_{SKL}(q \| \hat{p}) = -\frac{|I_l^u|}{|I_l^u| + |I_r^u|} D_{SKL}(I_l^u) - \frac{|I_r^u|}{|I_l^u| + |I_r^u|} D_{SKL}(I_r^u), \tag{6.18}$$

where I_l^u and I_r^u are the left and right subsets of unlabeled data that fall into this node, respectively.

Discussion The algorithm as introduced above leverages random forests to incorporate unlabeled data using priors similar as in the XR approach for boosting [Saffari et al., 2008]. The overall behavior of a randomized tree is a as follows: (i) minimize the impurity over labeled data through recursive splitting decision nodes and (ii) simultaneously minimize the cross entropy to a given prior q over the unlabeled data. Since usually $|\mathcal{X}_l| \ll |\mathcal{X}_u|$, it is clear that already after few splits the labeled samples are usually fully separated and for the rest of the tree-growing procedure the cross entropy is minimized; *i.e.*, the tree finally ends up learning the prior.

6.3 Experiments

In this section, we will give some detailed experimental analyses of the proposed DAS-RF method and will evaluate it on standard machine learning benchmarks and on object categorization. For all of the experiments, the main purpose is to proof the concept of the approach, analyze its empirical behavior and compare it to other SSL methods.

6.3.1 Machine Learning

The first experiment starts with an evaluation of DAS-RF on machine learning benchmarks. For all experiments, we set the $\alpha = 0.1$ and used 100 trees.

For fair comparison, we implemented the original random forest (RF) algorithm as proposed by Breiman [Breiman, 2001] and evaluated it within the same framework as our DAS-RF. Also SERBoost [Saffari et al., 2008] and RMSBoost [Saffari et al., 2009a] were evaluated in the same framework. For SVM and TSVM we used standard packages. We use the *g50c*, *Letter*, and *SensIt* datasets from the Semi-Supervised Benchmarks [Chapelle et al., 2006] and LibSVM repository [Chang and Lin, 2001]. A summary of these data sets is presented in Table 6.1. For *g50c*, we use the original splits. For the last two datasets, we

Dataset	# Train	# Test	# Class	# Feat.
g50c	50	500	2	50
Letter	15000	5000	26	16
SensIt (com)	78823	19705	3	100

Table 6.1: Data sets for the machine learning experiments.

Method	SVM	TSVM	SER	RMSB	RF	DAS-RF
g50c	91.7	93.1	91.9	**94.2**	89.1	<u>93.3</u>
Letter	70.3	65.9	76.5	**79.9**	76.4	<u>79.7</u>
SensIt	80.2	79.9	81.9	<u>83.7</u>	76.5	**84.3**

Table 6.2: Classification accuracy (in %) for machine learning datasets. DAS-RF stands for our method. We mark the best method bold-face and underline the second best.

randomly partition the original training set into two disjoint sets of labeled and unlabeled samples. We randomly select 5% of the training set to be labeled and assign the rest (95%) to the unlabeled set. We repeat this procedure 10 times and report the average classification accuracy in Table 6.2. As can be seen from this table, our method is always among the best two over these datasets with respect to other semi-supervised methods. Table 6.3 also shows the average computation time for these methods. It can also be seen, that DAS-RF is of course slower than the supervised methods, which comes from the fact that has to process 20 times more unlabeled data on the larger datasets. However, compared to the other semi-supervised methods, our method is faster in the presence of large amounts of data. Since our method is inherently parallel, we also implemented it on a GPU resulting in an *additional 3-times speed-up* compared to the CPU implementation.

Method	SVM	TSVM	SER	RMSB	RF	DAS-RF	GPU-DAS-RF
Letter	25	74	3124	125	35	72	29
SensIt	195	687	1158	514	125	410	137

Table 6.3: Computation (train+test) time (in seconds) for machine learning datasets. Compared to supervised RFs, our method is slower due to the iterative optimization over the unlabeled data but has the same speed during testing. Note that for the *g50c* data the computation times where similar for all algorithms.

Algorithm	$l = 15$	$l = 30$
RF	0.72	0.64
DAS-RF	0.70	0.60
LinSVM	0.74	0.65
improvement	2%	4%

Table 6.4: Comparison of RF and DAS-RF in terms of classification error over different numbers of labeled samples.

6.3.2 Object Categorization

For performing the categorization experiments, we chose the popular Caltech-101 dataset consisting of 101 object categories with between 31 and 800 labeled samples per category. Bosch *et al.* [Bosch et al., 2007] demonstrated state-of-the-art performance on that dataset using Random Forests. This dataset still provides a challenging benchmark for an inherently multi-class classifier.

In particular, for representation we use the L1-normalized PHOG [1] shape descriptors as introduced by Bosch *et al.* with 180 and 360 degrees, respectively. We trained RFs with 100 trees, using the information gain as node splitting criterion, ten random tests and a maximum tree depth of twenty. Additionally, in contrast to [Bosch et al., 2007], we train multi-class RFs. Please node that it is a much harder task to train a single multi-class classifier for 100 classes than 100 one-vs.-all classifiers. For training, we use a randomly chosen subset of labeled data and all other samples as unlabeled data. Also, note that in this work, we use much weaker representations and less engineering for the sake of speed and clarity than, for instance, compared to [Bosch et al., 2007].

For our SSL process, we allow a maximum of $m = 20$ iterations. For the cooling starting parameter we chose a simple exponential cooling function. We conduct our experiments with the typical amount of 15 and 30 labeled data, respectively. The final results are depicted in Table 6.4 while the improvement over the iterations is given in Figure 6.1. In these experiments, we also compare the results with the linear SVM for sanity check.

Binary Classification In the next experiment, we trained 100 1-vs.-all binary classifiers trained with 30 labeled samples and measured their improvements. In Table 6.5 we depicted the five best improving binary classifiers. Note that on all classes we got an average improvement of 33% and never observed an increasing error rate during training with DAS-RF. The reason why the improvements here are much better than compared to the multi-class experiment is that the latter problem is much more difficult.

[1] (10.3.2009) Available for download at http://www.robots.ox.ac.uk/ vgg/research/caltech/phog.html

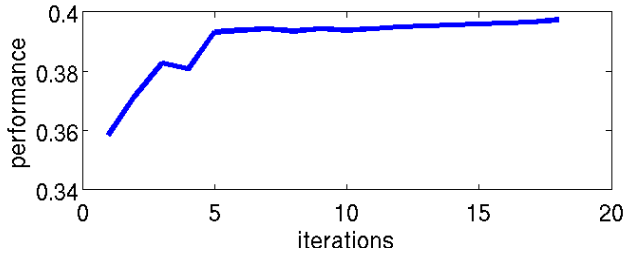


Figure 6.1: Improved average performance over the iterations for the multi-class problem.

Class	RF_{err}	DAS-RF_{err}	Relative Improvement
C4	0.0081	0.0033	58%
C5	0.0078	0.002	65%
C20	0.011	0.0013	87.5%
C33	0.007	0.003	52%
C81	0.0027	0.001	62.5%

Table 6.5: Comparison of RF with DAS-RF based on the binary classification error.

6.3.3 Airbag

The purpose of this experiment is to show two things: First, it shows that if unlabeled data does not help, the dynamics of the OOBE can be used as a safety mechanism and, second, that trivial self-training RFs do not succeed on a difficult multi-class categorization task. Hence, we trained two semi-supervised multi-class classifiers. However, while one was trained using the same settings as above the second one was trained on artificial corrupted unlabeled data. Additionally, we trained a RF performing self-learning on the (not corrupted) unlabeled data, *i.e.*, without using DA. The results are depicted in Figure 6.2. As can be seen, after 6 iterations the OOBE increases over a tolerance level from one iteration to the other one and we can automatically stop the training. As a result, we get the same performance as if only training on labeled data. The self-learning experiment fails even on not corrupted unlabeled data.

6.4 Summary

In this chapter, we introduced a novel semi-supervised learning method using random forests. We formulated the hidden labels of unlabeled samples as additional optimization variables and minimized the overall loss function using deterministic annealing. DA is a

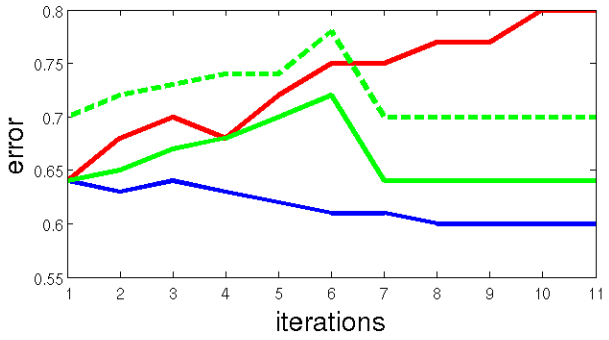


Figure 6.2: Training a DAS-RF on corrupted unlabeled data (green) and its OOBE (green dashed) and on not corrupted data (blue). After 6 iterations the SSL stops and it is trained only on labeled data. Self-learning is depicted in red.

fast non-convex optimization method based on random processes and thus fits perfectly into the nature of random forests. We further showed that random forests' out-of-bag-error can be used in order to measure if unlabeled data helps or not. In the experimental part, we evaluated our method on both machine learning benchmarks and object categorization on the Caltech-101 data set. In both experiments, we showed that DAS-RFs are able to robustly benefit from unlabeled data and solve multi-class SSL tasks.

Chapter 7

On-line Semi-Supervised Random Forests

In the previous chapter, we introduced a semi-supervised learning variant using randomized trees called DAS-RF. However, this algorithm is only suitable for off-line learning whereas we argued in previous chapters that in many applications on-line learners are required and especially for semi-supervised learning on-line methods are interesting due to their good scaling behavior compared to batch learners.

In this chapter, we will show that semi-supervised random forests based on DA-optimization can be extended to on-line learning. Semi-supervised RFs as introduced in the previous chapter use an off-line two-step optimization procedure, where in one step the objective function $\mathcal{F}$ is optimized and in the second step the distribution $\hat{\mathbf{p}}$ over the unlabeled samples, respectively. In order to modify the algorithm so that it is suitable for on-line learning, *i.e.*, both labeled and unlabeled samples arrive sequentially, one has to hence change both optimization steps to operate in on-line mode.

7.1 On-Line Random Forests

The original RF algorithm as proposed by Breiman [Breiman, 2001] is designed to learn in batch or off-line mode, *i.e.*, each tree is trained on a full sub-set of labeled samples drawn from $\mathcal{X}$. To make the algorithm operate in on-line mode, there exist two main questions that have to be answered:

1. How to perform bagging in on-line mode?

2. How to grow random trees on-the-fly?

7.1.1 On-Line Bagging

For the bagging part, we use the method proposed by Oza *et al.* [Oza, 2001] where the sequential arrival of the data is modeled by a Poisson distribution. Each tree $f_t(x)$ is updated on each sample k times in a row where k is a random number generated by Poisson(λ) and λ is usually set to a constant number, in our case equal to one. Oza proved convergence of this method to off-line bagging.

7.1.2 On-Line Random Decision Trees

Compared to on-line bagging, on-line learning of the decision trees is less trivial due to their recursive hard splitting nature which does not allow to correct errors further down the tree. There exist incremental methods for single decision trees but they are either memory intensive, because every node sees and stores all the data [Utgoff et al., 1997], or have to discard important information if parent nodes change. Some methods alleviate this problem by combining decision trees with ideas from neural networks [Basak, 2004] but have the disadvantage that they usually lose the $\mathcal{O}(\log n)$ evaluation time because samples are propagated to all nodes. In the following, we propose an on-line random forest algorithms that works by circumventing the pure recursive training of the trees by using a tree-growing procedure similar to evolving-trees [Pakkanen et al., 2004] or Hoeffding trees [Pfahringer et al., 2007].

Recall that in randomized decision trees, each decision node in a tree contains a test in form of $g(x) > \theta$. These tests usually contain two main parts: (i) a randomly generated test function, $g(x)$ which usually returns a scalar value and (ii) a threshold θ which based on the random feature decides the left/right propagation of samples. In off-line mode, RFs select randomly a set of such tests and then pick the best according to a quality measurement. If the threshold is also chosen randomly, the resulting RF is usually referred to *Extremely Randomized Forest* [Geurts et al., 2006].

In on-line mode, we grow extremely randomized trees by generating the test functions and thresholds randomly. During growing of a randomized tree, each decision node randomly creates a set of tests and picks the best according to a quality measurement such as the commonly used entropy or Gini. Computing such quality measures depends mainly on the estimation of the label densities, which can be performed in on-line mode.

More specifically, when a node is created it creates a set of N random tests $\mathcal{S} = \{(g_1(x), \theta_1), \ldots, (g_N(x), \theta_N)\}$. This node then starts to collect the statistics of the samples falling in it. It also maintains the statistics of the splits made with each test in $\mathcal{S}$. Denote by $\mathbf{p}_j = [p_1^j, \ldots, p_K^j]$ the statistics of class labels in node j. For a random test $s \in \mathcal{S}$, two sets of statistics are also collected: $\mathbf{p}_{jls} = [p_1^{jls}, \ldots, p_K^{jls}]$ and $\mathbf{p}_{jrs} = [p_1^{jrs}, \ldots, p_K^{jrs}]$ corresponding to the statistics of samples falling into left (l) and right (r) partitions ac-

cording to test s.

The gain with respect to a test s can be measured as:

$$\Delta L(\mathcal{R}_j, s) = L(\mathcal{R}_j) - \frac{|\mathcal{R}_{jls}|}{|\mathcal{R}_j|} L(\mathcal{R}_{jls}) - \frac{|\mathcal{R}_{jrs}|}{|\mathcal{R}_j|} L(\mathcal{R}_{jrs}), \tag{7.1}$$

where $\mathcal{R}_{jls}$ and $\mathcal{R}_{jrs}$ are the left and right partitions made by the test s and $|.|$ denotes the number of samples in a partition. Note that $\Delta L(\mathcal{R}_j, s) \geq 0$. A test with higher gain, produces better splits of the data with respect reducing the impurity of a node. Therefore, when splitting a node, the test with highest gain is chosen as the main decision test of that node.

When operating in the off-line mode, the decision node has access to all the data falling to that node, and therefore has a more robust estimate of these statistics, compared to a node operating in on-line mode. In the on-line mode, the statistics are gathered over time, therefore, the decision when to split depends on 1) if there has been enough samples in a node to have a robust statistics and, 2) if the splits are good enough for the classification purpose. Because, the statistics of the subsequent children nodes are based on this selection and since the errors in this stage cannot be corrected further down the tree when we already made a decision, we need to develop a method which can tell the node when it is appropriate to perform a split.

Therefore, we propose the following non-recursive strategy for on-line learning of the random decision trees: A newly generated tree starts with only one root node with a set of randomly selected tests. For each test in the node we gather the statistics on-line. We introduce two hyperparameters: 1) the minimum number of samples a node has to see before splitting α, 2) the minimum gain a split has to achieve β. Thus, a node splits when $|\mathcal{R}_j| > \alpha$ and $\exists s \in \mathcal{S} : \Delta L(\mathcal{R}_j, s) > \beta$.

After a split occured, we propagate the $\mathbf{p}_{jls}$ and $\mathbf{p}_{jrs}$ to the subsequent newly generated left and right leaf nodes, respectively. This way, a new node starts already with the knowledge of its parent nodes, and therefore, can also perform classification on-the-fly even without observing a new sample. The entire on-line RF algorithm is depicted in Algorithm 7.1.

Note that this tree-growing strategy is similar to that of evolving trees (ETrees) [Pakkanen et al., 2004]. An ETree is a tree-structed self-organizing map (SOM) used in many data analysis problems. In particular, in ETrees each node counts the number of observations seen so far and splits the node after a constant threshold has been exceeded. Another similar approach to ours is that of a Hoeffding tree [Pfahringer et al., 2007]. A Hoeffding tree is also a growing decision tree, where the split decision is made on the Hoeffding bound which theoretically guarantees that with probability $1-\rho$ the true statistical average of a random variable r is $\hat{r} - \epsilon$ with $\epsilon = \sqrt{\frac{ln(1/\rho)}{2n}}$, where n is the number of observations performed and $\hat{r}$ is the current estimate of the random variable.

Algorithm 7.1 On-Line Random Forests

Require: Sequential training example $\langle x, y \rangle$
Require: The size of the forest: T
Require: The minimum number of samples: α
Require: The minimum gain: β

// For all trees
for t from 1 to T **do**
    k $\leftarrow$ Poisson(λ)
    if $k > 0$ **then**
        // Update k times
        for u from 1 to k **do**
            $j =$ findLeaf(x).
            updateNode$(j, \langle x, y \rangle)$.
            if $|\mathcal{R}_j| > \alpha$ and $\exists s \in \mathcal{S} : \Delta L(\mathcal{R}_j, s) > \beta$ **then**
                Find the best test: $s_j = \arg\max_{s \in \mathcal{S}} \Delta L(\mathcal{R}_j, s)$.
                createLeftChild$(\mathbf{p}_{jls})$
                createRightChild$(\mathbf{p}_{jrs})$
            end if
        end for
    else
        Estimate $OOBE_t \leftarrow$ updateOOBE$(\langle x, y \rangle)$
    end if
end for
Output the forest $\mathcal{F}$.

Although both the ETree and the Hoeffding tree would definitely also be a useful choice for our splitting criterion, we believe that our approach, *i.e.*, continuously measuring the gain of a potential split, fits better to the inherent nature of decision trees.

Temporal Knowledge Weighting For some applications, such as tracking, the distribution of samples might change over time. Therefore, it is required to have temporal knowledge weighting that allows unlearning old information. If the algorithm is operating in such a scenario, we allow our forest to discard the entire tree. Note that the Poisson process of on-line bagging leaves out some trees from being trained on a sample. Therefore, we can estimate the $OOBE_t$ of each tree on-line. Based on this estimate, we propose to discard trees randomly from the ensemble where the probability of discarding a tree depends on its out-of-bag-error and also its age a_t (the number of samples it has seen so far). Since in an ensemble of trees the impact of a single tree is relatively low, discarding

one tree usually does not harm the performance of the entire forest. However, doing this continuously ensures adaptivity throughout time. This process is shown in Algorithm 7.2, where γ determines the temporal knowledge weighting rate. During the training of the forest, one tree is randomly chosen based on its age and if its $OOBE$ is large, then the tree is replaced with a new tree. In out tracking experiments, we fixed $\gamma = 0.05$.

Algorithm 7.2 Temporal Knowledge Weighting

Require: The knowledge weighting rate: γ

1: Select a tree randomly from $\{f_t | f_t \in \mathcal{F}, a_t > 1/\gamma\}$.
2: **if** $OOBE_t >$ rand() **then**
3: // Discard the tree.
4: $f_t =$ newTree()
5: **end if**

7.2 On-line Deterministic Annealing

Now that we know how to do on-line training of the randomized trees, we also have to perform the deterministic annealing on-line. This means we have to estimate $\hat{p}$ on-line by examining the sequentially arriving samples.

In fact, instead as in the off-line case where we estimated the set of distributions $\hat{\mathbf{p}} \in \mathcal{P}$ over all the unlabeled samples at once, we now find the best distribution for a single sample. Therefore, if a new sample $\mathbf{x}_i$ arrives, we initialize a new distribution $\hat{p}_i$ using the current confidence output of $\mathcal{F}_t$. Then, we iteratively apply the optimization of $\mathcal{F}_t$ and $\hat{p}_i$ only for the current sample $\mathbf{x}_i$ following the same two-step procedure and annealing schedule as in the off-line case. Note however, that during the successive decrease of the temperature parameter $T_0 > T_1 > \ldots > T_\infty = 0$, we all the time continue growing the tree in order to incorporate the intermediate steps. Since this steps might be suboptimal, however, we keep only the tree state for $T = 0$ and discard all the others. Afterwords, $\mathbf{x}_i$ is discarded and the training proceeds with the next sample $\mathbf{x}_{i+1}$. We depict the algorithm in Algorithm 7.3.

7.3 Summary

In this chapter, we have extended the learning principle of DAS-RF to on-line learning. We showed that therefore, basically, two steps are necessary: the random forests have to be on-line learning capable as well as the deterministic annealing has to be performed on-line. We presented a novel on-line formulation of random forests that circumvents

Algorithm 7.3 On-Line Semi-supervised Random Forests

Require: Sequential training example $\langle x, y \rangle$
Require: The size of the forest: T
Require: The minimum number of samples: α
Require: The minimum gain: β
Require: A starting heat parameter T_0 and a cooling function $c(T, m)$

// Labeled update
if $y \in K$ **then**
    $\mathcal{F}_t \leftarrow$ update($\mathcal{F}_{t-1}, \mathbf{x}, y$)
else
    // Unlabeled update
    Set the epoch: $m = 0$.
    repeat
        Get the temperature: $T_{m+1} \leftarrow c(T_m, m)$.
        Set $m \leftarrow m + 1$.
        Compute $p^*(k|\mathbf{x})$.
        // For all trees
        for n from 1 to N **do**
            Draw a random label, $\hat{y}$ from $p^*(\cdot|\mathbf{x})$ distribution.
            Update the tree: $f_n^t \leftarrow$ updateTree($f_n^{t-1}, \mathbf{x}, \hat{y}$).
        end for
    until Stopping condition
end if
Output the forest $\mathcal{F}$.

the recursive split nature of decision trees by letting them learn using a tree-growing mechanism. DA can be made on-line by applying the annealing schedule on a single sample.

Chapter 8

Multiple Instance Learning with Random Forests

Semi-supervised learning algorithms have to learn from ambiguously labeled samples because the true labels of unlabeled samples are unknown. In machine learning, there exist a second learning paradigm called *multiple-instance learning* (MIL) [Keeler et al., 1990, Dietterich et al., 1997] which is very similar to SSL and also has to resolve ambiguities during the learning process. In particular, in multiple-instance learning, training samples are provided in form of bags, where each bag consists of several instances. Labels are only provided for the bags and not for the instances. The labels of instances inside positive bags are unknown, but it is guaranteed that at least one instance has a positive label. Contrary, in negative bags all instances can be considered as being negative. (See also Figure 8.1 for an illustration of the principle.)

In this chapter, we present a multiple-instance learning algorithm based on random forests. We thus call the method *MILForests*. MILForests bring the advantages of random forests – *i.e.*, speed, multi-class capability, multi-processing, noise resistance, *etc.*– to multiple-instance learning, where usually different methods have been applied. In turn, extending random forests in order to allow for multiple-instance learning allows vision tasks where RFs are typically applied to benefit from the flexibility of MIL. MILForests are very similar to conventional random forests. However, since the training data is provided in form of bags, during learning the real class labels of instances inside bags are unknown. In the following, we will show that multiple-instance learning is a special case of SSL because all instances inside negative bags can be considered as labeled samples and all instances inside positive bags as unlabeled, respectively. Based on this insight we can use a similar optimization approach as for the semi-supervised RFs, *i.e.*, deterministic annealing, in order to find the hidden class labels of instances in positive bags.

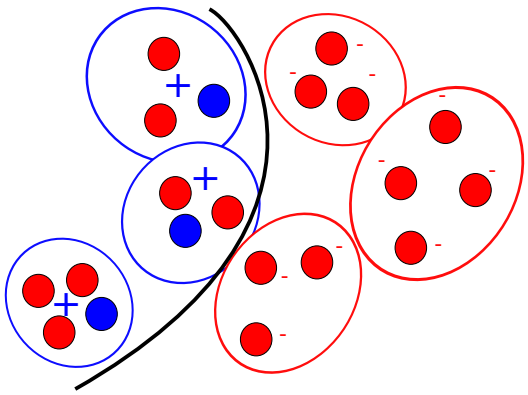


Figure 8.1: Multiple instance learning principle: Positive bags (blue regions) consist of both positive and negative instances; however, the "real" instance labels are unknown to the learner. By contrast, in negative bags (red) all instances are guaranteed to be negative.

8.1 Related Work

As we have seen in Chapter 2, in traditional supervised learning training data is provided in form of $\{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is an instance and, in the binary case, $y_i \in \{-1, +1\}$ the corresponding label. In multiple instance learning training samples are given in bags $\{(B_1, y_1), \ldots, (B_n, y_n)\}$, where each bag may consist of an arbitrary number of instances, *i.e.*, $B_i = \{x_i^1, x_i^2, \ldots, x_i^{n_i}\}$. Negative bags B_i^- consist of only negative instances. Ambiguity is introduced into learning by the constraint that for positive bags B_i^+, it is only guaranteed that there exist at least one positive instance (also called *witness* of the bag). There is no information about other instances in the bag. In fact, they might not even belong to the negative class. The task is to learn either a bag classifier $f : B \to \{-1, 1\}$ or an instance classifier $f : \mathbb{R}^d \to \{-1, 1\}$. However, bag classification can be obtained automatically from instance classification, *e.g.*, by using the *max* operator $p_i = \max_j \{p_{ij}\}$ over posterior probability estimates p_{ij} for the j-th instance of the i-th bag.

There exists a vast amount of literature and many different approaches on how to solve the MIL problem. Here, we briefly review some of the most popular ones. The most naïve approach is to simply ignore the MIL setting and train a supervised classifier on all instances with the bag label. Blum and Kalai [Blum and Kalai, 1998], for instance, showed that one can achieve reasonable results when training an instance classifier that is robust to class label noise. As we will show later in the experimental part, RFs are also promising candidates for such a naïve approach.

Many MIL methods work by adapting supervised learners to the MIL constraints,

mostly using SVM-type learners. For example, Andrews *et al.* [Andrews et al., 2003] proposed two different types of SVM-MIL approaches mi-SVM and MI-SVM. They differ basically on their assumptions, *i.e.*, the first method tries to identify the labels of all instances in a bag while the latter one finds only the witness and ignores all others. Another SVM-based approach MICA [Mangasarian and Wild, 2005] tries to find the witness using linear programming.

There also exist some boosting-based methods, *e.g.*, [Viola et al., 2006]. Wang and Zucker [Wang and Zucker, 2000] trained a nearest neighbor algorithm using Hausdorff distance. Other popular approaches are based on the diverse-density assumption, for example [Maron and Lozano-Perez, 1997, Zhang and Goldman, 2001], which more directly tries to address the MIL problem via finding a more appropriate feature representation for bags. In MILES, Chen *et al.* [Chen et al., 2006, Foulds and Frank, 2008] trained a supervised SVM on data mapped into a new feature space based on bag similarities. There exist also approaches for training decision trees in a MIL fashion, *e.g.*, [Blockeel et al., 2005].

While multiple-instance learning has been used in many applications such as text-categorization [Andrews et al., 2003], drug activity recognition [Dietterich et al., 1997] or computer security problems [Ruffo, 2000], especially computer vision is one of the most important domains where multiple instance-learning algorithms are recently applied, because in practice data is often provided in a similar manner. For example, in case of object detection bounding boxes are usually cropped around the target object and provided as positive training samples. The decision where exactly to crop the object and at which size is up to the human labeler and it is often not clear if those patches are best suited for the learner. Additionally, it would also ease the labeling effort if the exact object position has not to be labeled. Hence, it would be desired to provide the learner only a rough position of the positive object and leave it on its own how to incorporate the information in order to deliver best classification results. For standard supervised learning techniques it is hard to resolve such ambiguously labeled data. Applying MIL in this example, the rough object position would correspond to a bag and patches inside the bag to instances. During training, MIL would find those patches that lead to best classification results and leave out the others. Furthermore, many authors applied MIL to image retrieval [Zhang and Goldman, 2002, Vijayanarasimhan S. Grauman, 2008] or image categorization tasks [Chen et al., 2006]. Another computer vision application where multiple-instance learning can be used is to tackle the alignment problem when training appearance-based detectors based on boosting [Viola et al., 2006], speed-up classifier cascades [Zhang and Viola, 2008] or even action recognition [Stikic and Schiele, 2009]. In case of object tracking, it is mostly hard to decide which patches to use for updating the adaptive appearance model. If the tracker location is not precise, errors may accumulate

which finally leads to drifting. Recently, Babenko *et al.* [Babenko et al., 2009b] demonstrated that using MIL for tracking leads to much more stable results. For most of these vision tasks SVM variants or boosting have been used.

8.2 Multiple Instance Learning as a special case of Semi-supervised Learning

At the first glance, multiple instance learning and SSL do not seem to have many things in common. In the literature, they are hence considered as being two different branches of machine learning. However, it can be easily shown [Zhou and Xu, 2007] that multiple-instance learning can be seen as a special case of semi-supervised learning, however, with having an additional constraint on some parts of the unlabeled data and without knowing one single real positive instance.

More formally, consider a training set consisting of bags $\{(B_1, y_1), \ldots, (B_n, y_n)\}$ and assume p positive and q negative bags, with $p + q = n$. The negative bags are ordered before the positive bags, *i.e.*, $\{(B_1^-, y_1), (B_2^-, y_2) \ldots, (B_{n-1}^+, y_{n-1}), (B_n^+, y_n)\}$. If we then take the instances bag-by-bag into an instance set $\{(\mathbf{x}_{1,1}, y_{1,1}), (\mathbf{x}_{1,2}, y_{1,2}), \ldots, \ldots, (\mathbf{x}_{n,n_i-1}, y_{n,n_i-1}), (\mathbf{x}_{n,n_i}, y_{n,n_i})\}$ it can easily be seen that the first $Q = \sum_{i=1}^{q} n_i$ instances are from negative bags and the remaining $P = \sum_{i=q+1}^{n} n_i$ are from positive bags. Now, we can denote $\mathcal{X}_l = Q$ and $\mathcal{X}_u = P$ *subject to* $\forall i \in \mathcal{X}_u : \sum_{j=1}^{n_i} \mathbb{I}(y_j = 1) \geq 1$. If we interpret $\mathcal{X}_l$ as labeled data set with $y_i = -1$ and $\mathcal{X}_u$ as unlabeled data set, we can easily see that MIL is a special case of semi-supervised learning, however, having the additional constraint that for a sub-sequence of instances $\{\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n_i}\}$ coming from the same positive bag B_i^+ at least one instance is positive. Note that MIL can hence also be interpreted as an *asymmetric* semi-supervised learning problem, because latency is only given for the positive bags and the instances therein. Also based on this insight, in the following section we will present a multiple-instance learning method using random forests.

8.3 MILForests

In the following, we introduce a novel multiple instance learning algorithm using randomized trees called *MILForests*. MILForests deliver multi-class instance classifiers in form of $F(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{Y} = \{1, \ldots, K\}$. Formally, in contrast to the binary case, for multi-class MIL problems the data is provided in form of $\{(B_1, y_1), \ldots, (B_n, y_n)\}$, where $y_i \in \{1, \ldots, K\}$. This means that for all bags the instance labels are unknown and can consist of labels $\{1, \ldots, K\}$. It is only guaranteed that at least one instance has the bag

label. This makes MILForests different to most previous MIL algorithms that only yield binary classifiers and require to handle a multi-class problem by a sequence of binary ones.

One obvious way to design RFs capable of solving MIL tasks is to adopt MIL versions for single decision trees [Blockeel et al., 2005]. However, strategies developed for common decision trees are hard to apply for RFs due to the random split nature of their trees. For example, improper regularization of trees of a RF on the node level can decrease the diversity $\bar{\rho}$ among trees and thus increase the overall generalization error (see Eq. (2.16)). Additionally, the method proposed in [Blockeel et al., 2005] is based on simple heuristics and needs a complicated inter-node communication channel. Thus, in order to perform multiple instance learning with random forests one has to find an optimization strategy that preserves the diversity among the trees. In fact, this is a similar condition as for SSL with random forests. Hence, following this condition and the arguments stated in the previous section 8.2, we makes sense to use a similar optimization strategy as for our semi-supervised random forests introduced in Chapter 6.

Therefore, we formulate multiple instance learning as an optimization procedure where the labels of the instances become the optimization variables. The algorithm tries to uncover the true labels of the instances in an iterative manner. Given such labels, one can train a supervised classifier which then can be used to classify both instances and bags.

Let $B_i, i = 1, \ldots, n$ denote the i-th bag in the training set with label y_i. Each bag consists of n_i instances: $\{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^{n_i}\}$. We write the objective function to optimize as:

$$\begin{aligned}(\{y_i^j\}^*, F^*) =& \arg\min_{\{y_i^j\}, F(\cdot)} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \ell(F_{y_i^j}(\mathbf{x}_i^j)) \\ & \text{s.t. } \forall i : \sum_{j=1}^{n_i} \mathbb{I}(y_i = \arg\max_{k \in \mathcal{Y}} F_k(\mathbf{x}_i^j)) \geq 1.\end{aligned} \tag{8.1}$$

The objective in this optimization procedure is to minimize a loss function $\ell(\cdot)$ which is defined over the entire set of instances by considering the condition that at least one instance in each bag has to be from the target class. Note that $\mathbb{I}(\cdot)$ is an indicator function and $F_k(\mathbf{x})$ is the confidence of the classifier for the k-th class, *i.e.*, $F_k(\mathbf{x}) = p(k|\mathbf{x}) - \frac{1}{K}$. Often, the loss function depends on the classification margin of an instance. In the case of Random Forests, the margin can be written as [Breiman, 2001]

$$m(\mathbf{x}, y) = p(y|\mathbf{x}) - \max_{\substack{k \in \mathcal{Y} \\ k \neq y}} p(k|\mathbf{x}) = F_y(\mathbf{x}) - \max_{\substack{k \in \mathcal{Y} \\ k \neq y}} F_k(\mathbf{x}). \tag{8.2}$$

Note that for a correct classification $m(\mathbf{x}, y) > 0$ should hold. Overall, it can easy be seen that Eq. (8.1) is a non-convex optimization problem because a random forest has

to be trained and simultaneously a suitable set of labels y_i^j has to be found. Due to the integer values of the labels y_i^j, this problem is a type of integer programming and is usually difficult to solve. In order to solve this non-convex optimization problem without loosing too much of the training speed of random forests, we will need a fast optimization procedure. As we have seen in the previous chapter for SSL, deterministic annealing is an ideal candidate for such a task and we will hence use it also here for solving our MIL-constrained learning task.

8.3.1 Optimization

In order to optimize our MIL objective function (Eq. (8.1)), we propose the following iterative strategy: In the first iteration, we train a naïve RF that ignores the MIL constraint and uses the corresponding bag labels for instances inside that bag. Then, after the first iteration, we treat the instance labels in positive bags as random binary variables. These random variables are defined over a space of probability distributions $\mathcal{P}$. We now search a distribution $\hat{\mathbf{p}} \in \mathcal{P}$ for each bag which solves our optimization problem in Eq. (8.1). Based on $\hat{\mathbf{p}}$ each tree randomly selects the instance labels for training. Hence, based on the optimization of $\hat{\mathbf{p}}$ we try to identify the real but hidden labels of instances.

We reformulate the objective function given in Eq. (8.1) so that it is suitable for DA optimization

$$\mathcal{L}_{DA}(F, \hat{\mathbf{p}}) = \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{k=1}^{K} \hat{p}(k|\mathbf{x}_i^j) \ell(F_k(\mathbf{x}_i^j)) + T \sum_{i=1}^{n} H(\hat{\mathbf{p}}_i), \tag{8.3}$$

where T is the temperature parameter and

$$H(\hat{\mathbf{p}}_i) = -\sum_{j=1}^{n_i} \sum_{k=1}^{K} \hat{p}(k|\mathbf{x}_i^j) \log(\hat{p}(k|\mathbf{x}_i^j)) \tag{8.4}$$

is the entropy over the predicted distribution inside a bag. It can be seen that the parameter T steers the importance between the original objective function and the entropy. If T is high, the entropy dominates the loss function and the problem can be easier solved due to the convexity. If $T = 0$ the original loss dominates (Eq. (8.1)). Hence, DA first solves the easy task of entropy minimization and then by continuously decreasing T from high values to zero gradually solves the original optimization problem, *i.e.*, finding the real but hidden instance labels y and simultaneously training an instance classifier.

In more detail, for a given temperature level, the learning problem can be written as

$$(F^*, \hat{\mathbf{p}}^*) = \arg\min_{\hat{\mathbf{p}}, F(\cdot)} \mathcal{L}_{DA}(F, \hat{\mathbf{p}}) \quad \text{s.t. } \forall i : \sum_{j=1}^{n_i} \mathbb{I}(y_i = \arg\max_{k \in \mathcal{Y}} F_k(\mathbf{x}_i^j)) \geq 1. \tag{8.5}$$

We split this optimization problem up into a two-step convex optimization problem analog to an alternating coordinate descent approach. In the first step, the objective function $\mathcal{F}$ is optimized by fixing the distribution $\hat{\mathbf{p}}$ and optimizing the learning model. In the second step, the distribution p^* over the bags according to the current entropy level is adjusted. Note that both individual steps are convex optimization problems.

The detailed optimization now runs analogue to the SSL case; *i.e.*, for a given distribution over the bag samples, we randomly choose a label according to $\hat{\mathbf{p}}$. We repeat this process independently for every tree f in the forest. Let $\{\hat{y}_{ij}\}$ be the randomly drawn labels according to the distribution $\hat{\mathbf{p}}$ for t-th tree. The optimization problem for the t-th tree becomes

$$f_t^* = \arg\min_{f} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \ell(f_{\hat{y}_i^j}(\mathbf{x}_i^j)) \quad \text{s.t. } \forall i : \sum_{j=1}^{n_i} \mathbb{I}(y_i = \arg\max_{k \in \mathcal{Y}} f_k(\mathbf{x}_i^j)) \geq 1. \tag{8.6}$$

Since the margin maximizing loss function is convex, this loss function is also convex. In order to not violate the MIL constraint, after having randomly selected instance labels for a bag, we always set the instance with the highest probability according to $\hat{\mathbf{p}}$ equal to the bag label. At this stage we train all the trees in the forest by the formulation given above.

After we trained the random forest, we enter the second stage where we find the optimal distribution according to

$$\hat{\mathbf{p}}^* = \arg\min_{\hat{\mathbf{p}}} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{k=1}^{K} \hat{p}(k|\mathbf{x}_i^j) \ell(F_k(\mathbf{x}_i^j)) + T \sum_{i=1}^{n} H(\hat{\mathbf{p}}_i). \tag{8.7}$$

The optimal distribution is found by taking the derivative *w.r.t* p and setting it to zero. We depict all detailed steps of the method in Algorithm 8.1.

8.4 On-line MILForests

Although there have been proposed numerous approaches to the MIL problem, most of them operate in off-line or batch mode. In practice, however, it would also be desired

Algorithm 8.1 MILForests

Require: Bags $\{\mathcal{B}_i\}$
Require: The size of the forest: T
Require: A starting heat parameter T_0
Require: An ending parameter T_{min}
Require: A cooling function $c(T, m)$
Set: $\forall i : \hat{y}_i^j = y_i$
Train the RF: $\mathcal{F} \leftarrow$ trainRF($\{\hat{y}_i^j\}$).
Init epochs: $m = 0$.
while $T_{m+1} \geq T_{min}$ **do**
    Get the temperature: $T_{m+1} \leftarrow c(T_m, m)$.
    Set $m \leftarrow m + 1$.
    $\forall \mathbf{x}_i^j \in \mathcal{B}_i, k \in \mathcal{Y}$: Compute $p^*(k|\mathbf{x}_i^j)$
    for t from 1 to T **do**
        $\forall \mathbf{x}_i^j \in \mathcal{B}_i$: Select random label, $\hat{y}_i^j$ according to $p^*(\cdot|\mathbf{x}_i^j)$
        Set the label for instance with highest $p^*(\cdot|\mathbf{x}_i^j)$ equal to bag label
        Re-train the tree:
        $f_t \leftarrow$ trainTree($\{\hat{y}_i^j\}$).
    end for
end while
Output the forest $\mathcal{F}$.

to apply MIL in scenarios where we have limited access to the problem domain due to dynamic environments or streaming data sources. In the following, we hence take this considerations into account and show how MILForests can be extended to on-line learning.

MILForest as introduced above are trained off-line using a two-step optimization procedure Eq. (8.3), where in one step the objective function $\mathcal{F}$ is optimized and in the second step the distribution $\hat{\mathbf{p}}$ over the bags, respectively. In order to modify the algorithm so that it is suitable for on-line learning, *i.e.*, the bags B_i arrive sequentially, one has to change both optimization steps to operate in on-line mode. As we have seen in the previous chapter, randomized decision trees can be trained on-line using a tree-growing scheme.

Besides on-line training of the randomized trees, we also have to perform the deterministic annealing on-line. This means we have to estimate $\hat{p}$ on-line by examining the sequentially arriving samples. For calculating DA on-line we can make use of the same principle as was shown in the previous chapter, except the fact that in MIL we do not assume the individual instances to arrive sequentially but the bags. Therefore, if a new bag B_i arrives, we initialize a new distribution $\hat{p}_i$ over its instances using the current con-

fidence output of $\mathcal{F}_t$. Then, we iteratively apply the optimization of $\mathcal{F}_t$ and $\hat{p}_i$ only for the current bag B_i following the same two-step procedure and annealing schedule as in the off-line case (Eq. (8.6),Eq. (8.7)). Afterwards, B_i is discarded and the training proceeds with the next bag B_{i+1}.

8.5 Experiments

In the following we will evaluate the proposed algorithm on standard MIL machine learning benchmark datasets. We also evaluated the on-line extension of MILForests, however, depict the results in the final experimental Chapter 9 along with other competitive on-line learning algorithms. Note that, in general, we abstain from any data set or feature engineering procedures, since the main purpose is to compare the different learning methods.

8.5.1 Benchmark Datasets

We first evaluate our proposed MILForests on popular benchmark datasets used in most studies of multiple-instance learning algorithms, *i.e.*, the *Musk1 and Musk2* drug activity datasets proposed by Dietterich [Dietterich et al., 1997] and the *Tiger, Elephant and Fox* image datasets proposed by Andrews *et al.* [Andrews et al., 2003][1]. For sanity check we also tested common random forests [Breiman, 2001], *i.e.*, ignoring the MIL constraint. For all learners we used 50 trees with a maximum depth of 20. As cooling schedule we used a simple exponential function in form of $T_t = e^{-t \cdot C}$, where t is the current iteration and the constant $C = \frac{1}{2}$.

As can be observed, the performance of the individual approaches varies highly depending on the data set. The experiments show that MILForests achieve state-of-the-art performance and are even outperforming several SVM-based approaches and those based on boosting. Especially for the vision problems, we are always among the best. Also the naïve RF approach yields surprisingly good performance, especially on *Fox* and *Musk1*; however, it cannot catch up with the performance of its MILForest counterpart. One explanation for this might be that RFs are less susceptible to noise compared to other learning methods, which is necessary for the naïve approach [Blum and Kalai, 1998]. Compared to its most similar SVM variant (AL-SVM), MILForest outperforms it on two datasets, draws on one and performs worse on two. Finally, it has to be mentioned that especially for [Gehler and Chapelle, 2007] and [Bunescu and Mooney, 2007] better results can be achieved by incorporating prior knowledge into the learners, *e.g.*, how many "real" positives exist inside bags; which however also holds for MILForests.

[1] Note that for repeatability we will make our C++ implementation available on-line.

Method	Elephant	Fox	Tiger	Musk1	Musk2
RandomForest [Breiman, 2001]	74	60	77	85	78
MILForest	**84**	**64**	82	85	82
MI-Kernel [Andrews et al., 2003]	**84**	60	**84**	**88**	**89**
MI-SVM [Zhou et al., 2009]	81	59	**84**	78	84
mi-SVM [Zhou et al., 2009]	82	58	79	87	84
MILES [Chen et al., 2006]	81	62	80	**88**	83
SIL-SVM [Bunescu and Mooney, 2007]	**85**	53	77	**88**	**87**
AW-SVM [Gehler and Chapelle, 2007]	82	**64**	83	86	84
AL-SVM [Gehler and Chapelle, 2007]	79	63	78	86	83
EM-DD [Zhang and Goldman, 2001]	78	56	72	85	85
MILBoost-NOR [Viola et al., 2006]	73	58	56	71	61

Table 8.1: Results and comparisons in terms of percent classification accuracy on popular MIL benchmark datasets. We mark either of the two best performing methods bold face.

Method	Corel-1000	Corel-2000		Testing$_{[sec.]}$	Training$_{[sec.]}$
MILForest	59	66		4.6	22.0
MILES	58	67		180	960

Table 8.2: Results and comparisons on the COREL image categorization benchmark. Additionally, we put the training and testing times in seconds.

8.5.2 Corel Dataset

Here, we evaluate our proposed methods on the Corel-1000 and Corel-2000 image dataset for region-based image classification. The data set consists of 2000 images with 20 different categories. Each image is a bag consisting of instances obtained via oversegmentation. It is thus a typical MIL problem. In order to allow for fair comparison we used the same data settings and features as proposed by Chen *et al.* [Chen et al., 2006]. For the results we used the same settings as in our previous experiments. In contrast to most other approaches, we did not train 20 1-vs.-all classifiers, but trained one multi-class forest, which is usually a more difficult task. We compare MILForests with MILES, the original algorithm proposed on this data set [Chen et al., 2006]. Since MILES is a binary algorithm we trained 20 1-vs.-all MILES classifiers and depict the results in Table 8.2. As can be seen, MILForests achieve competitive results for multi-class scenarios, however, being much faster. We measured the average time on a standard Core Duo machine with 2.4 Ghz.

Figure 8.2: Some samples of the 20 COREL categories and their corresponding segmentations [Chen et al., 2006].

8.6 Summary

In this chapter, we reviewed multiple instance learning as another import machine learning paradigm that has tight relations to SSL. In fact, we showed that MIL can be treated as special case of SSL, however, with the difference that unlabeled data coming from the same bag are constrained that at least one instance should have the bag label. Following from this insight, we showed how we can use DA in order to train multiple instance random forests. In the experiments, we demonstrated that MILForests achieve state-of-the-art results on benchmark data sets while being faster than competitive methods and inherently multi-class. In Chapter 9, we will evaluate the on-line version on the task of visual object tracking.

Chapter 9

Visual Object Tracking

In this chapter, we will apply and test the proposed on-line learning methods of this thesis on one of the cardinal problems in computer vision, *i.e.*, visual object tracking. Tracking of a priori unknown objects and object types in 2D image space is one of the biggest challenges in computer vision. In detail, the goal of tracking is to continuously try to locate a target region in a video by estimating the relative displacement between successive frames when either the camera and/or the object is moving. Despite the huge amount of research spent on this task it is still hard to design robust tracking systems that can perform similar to humans. Visual trackers have to cope with all variations that occur in natural scenes such as shape and appearance changes, different illuminations as well as varying poses or partial occlusions.

According to [Comaniciu et al., 2003], the process of tracking can be divided into two major components: (i) filtering and association and (ii) target representation and localization. The first class of approaches tries to exploit motion information of the target from previously seen frames and predicts future object locations. For these kind of approaches usually filtering techniques such as Kalman filters [Kalman, 2008] or particle filters [Li et al., 2007] are used. This is also related to the problem of data association [Bar-Shalom, 2008], which models the correspondences of multiple target candidates. The second approach, target representation and localization, ignores temporal information as well as object dynamics and, as the name suggests, more deals with finding good representations for the target object, which are then used for localization. One of the most popular methods within this field is global template-based tracking [Avidan, 2007], where the main idea is to define some error function based on the pixel intensities for some target image. For instance, often the Normalized-Cross Correlation (NCC) is used to compare the given template with the target image

$$c(u,v) = \frac{1}{n}\sum_{x,y}\frac{(f(x+u,y+v)-\bar{f})\cdot(t(x,y)-\bar{t})}{\sigma_f \cdot \sigma_t}, \tag{9.1}$$

where $t(x, y)$ is the image function of the template and $f(x+u, y+v)$ is the sub-patch of the input image at location (u, v). $\bar{t}$ and $\bar{f}$ are the mean values of the template and the sub-patch, σ_t and σ_f are the corresponding standard deviations, respectively. n is the number of pixels in the template. Note that for calculating the mean $\bar{f}$ and standard deviations σ_f one can use integral data structures in order to drastically speed-up the calculations of the pixel sums [1]. Template matching is not limited to simple raw pixel correspondences and can also be extended to higher-level representations, such as histograms of oriented gradients (HOGs) [Adam et al., 2006]. In order to model simple or complex motions in template tracking usually parametric models can also be incorporated [Shi and Tomasi, 1994, Baker and Matthews, 2004]. See also [Yilmaz et al., 2006] for a more detailed review.

9.1 Tracking as a discriminative Classification Problem

A recently dominating trend in tracking is to apply appearance-based classifiers in order to track objects because they are able to deliver highly accurate results in real-time speeds. Such tracking-by-detection systems [Liu and Leordeanu, 2005, Avidan, 2007] usually train a classifier at the very first frame versus its local background and perform re-detection in the succeeding frames. In order to handle rapid appearance and illumination changes, recent works, *e.g.*, [Grabner and Bischof, 2006] use on-line classifiers that perform self-updating on the target object. Such on-line classifiers are usually highly accurate and fast since they only have to discriminate the object from its current local background. Figure 9.1 illustrates the tracking approach as proposed in [Grabner and Bischof, 2006]. As can be seen, the process starts by marking the target object in frame $t = 0$. An initial classifier $F_0(\mathbf{x})$ is then trained based on $\mathcal{X}_0 = \mathcal{X}_0^+ \cup \mathcal{X}_0^-$, where $\mathcal{X}_0^+$ corresponds to one patch, the marked target object, and $\mathcal{X}_0^-$ corresponds to surrounding negative patches. At frame $t = t + 1$ the trained classifier is applied in a sliding window manner in a predefined local search region in order to re-detect the marked object. The actual detection is positioned at the peak of the estimated confidence map. Then, similar to frame $t = 0$, $F_0(\mathbf{x})$ is on-line updated with $\mathcal{X}_{t+1} = \mathcal{X}_{t+1}^+ \cup \mathcal{X}_{t+1}^-$, where $\mathcal{X}_{t+1}^+$ corresponds to the estimated location of the target object, and $\mathcal{X}_{t+1}^-$ corresponds to surrounding patches of the estimated detection. Since this tracking process runs in loops, we also talk about a "tracking loop".

Although the approach of Grabner and Bischof has been shown to yield fast and highly accurate trackers, its main drawback lies in the fact that it can easily drift in case of wrong updates during learning [Matthews et al., 2004]. This comes from the fact that

[1] www.idiom.com/ zilla/Papers/nvisionInterface/nip.html (14.5.2010)

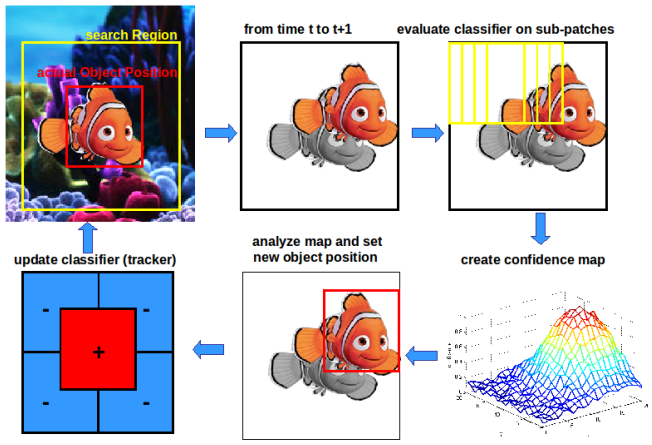


Figure 9.1: The original on-line AdaBoost tracking loop as proposed by [Grabner and Bischof, 2006].

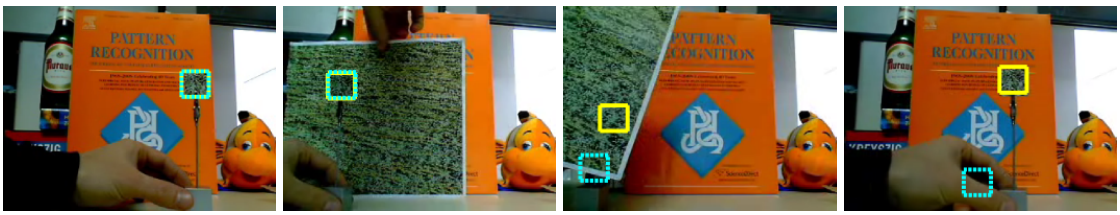


Figure 9.2: Tracking of a textured patch with difficult background (same texture). As soon as the object becomes occluded the original tracker from [Grabner and Bischof, 2006] (dotted cyan), drifts away. Our proposed methods (yellow) successfully re-detects the object and continues tracking.

the approach performs self-learning; *i.e.*, the tracker relies only on its own predictions. Yet, during tracking it is hard to decide where to select the positive and negative updates necessary for self-updating. As we have seen above, usually simple heuristics are used where positive updates are taken at the peak of the confidence map and negative updates from low-confident regions. If the update patches are selected wrongly due to a wrong confidence map, errors can be accumulated over time ending up in learning wrong things. Additionally, if the object is in principle re-detected correctly but the alignment is not perfect also slightly wrong updates are generated (*a.k.a. label jitter*) and can lead to drifting. Figure 9.2 further illustrates the problem.

9.2 An one-shot semi-supervised learning formulation for tracking

Tracking-by-detection, in principle, is an ill-posed problem. The complicated task in tracking using an appearance-based on-line classifier is to continuously apply self-training while avoiding wrong updates that my cause drifting. As already discussed above, the problem with these approaches is that the self-updating process may easily cause drifting in case of wrong updates. Even worse, the tracking-by-detection approach suffers also from the fact that usually on-line counterparts of supervised learning algorithms are used, which are not designed for handling ambiguity of class labels; for example, despite the fact that boosting is known to by highly susceptible to label noise – as we have seen in Section 5.2 – it is widely used in self-learning based tracking methods. This severe problem of adaptive tracking-by-detection methods can also be explained by the exploration-exploitation problem or the stability-plasticity dilemma [Grossberg, 1998]: If the classifier is trained only with the first frame, it is less error-prone to occlusions and can virtually not drift. However, non adaptive classifiers are not able to follow an object undergoing rapid appearance and viewpoint changes. On the other hand, on-line classifiers that perform self-learning on their confidence maps are highly adaptive but easily drift in case of wrong updates.

As can be easily observed, the only time when the classifier can assume having correct labels is at frame $t = 0$. During ongoing tracking, the classifier observes exclusively unlabeled samples and can only rely on its own believes. Hence, from a learning perspective, we have to solve a learning task where the individual samples arrive sequentially, are only labeled at the beginning and the rest of the time unlabeled, respectively. We thus state the following proposition:

Tracking-by-detection is an one-shot semi-supervised learning problem!

Following this observation, we further argue that one should apply semi-supervised learning methods rather than supervised ones, which is more intuitive. Hence, in Figure 9.4 we present a modified tracking loop where labeled data exist only in the first frame. In all subsequent frames t with $t = 1, \cdots, \infty$, we exploit subpatches as unlabeled samples. Since this is a classical semi-supervised learning formulation, we can use one of the semi-supervised boosting approaches (*i.e.*, on-line SemiBoost and on-line SERBoost) or semi-supervised random forests, introduced in this thesis as learners. As the semi-supervised boosting approaches need a prior classifier that "guides" the on-line learner

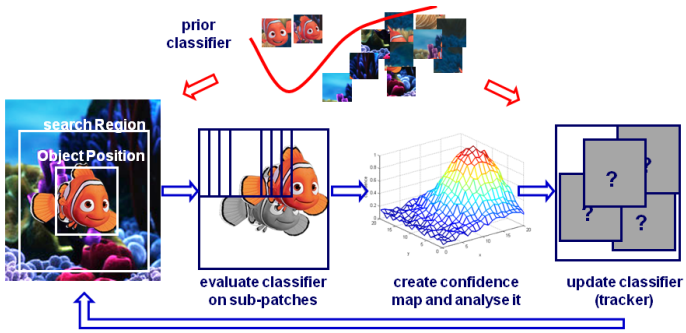


Figure 9.3: The semi-supervised tracking loop: as can be seen, the on-line classifier is "aware" that putative update patches are unlabeled data, which are incorporated using prior knowledge in form of a static classifier.

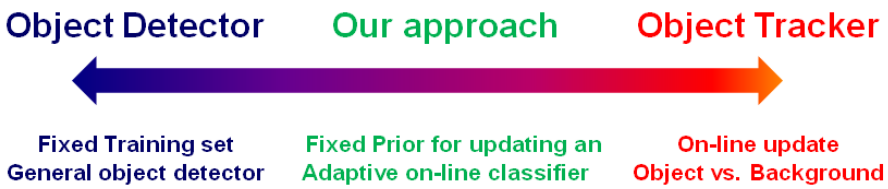


Figure 9.4: Detection and tracking in principle can be viewed as the same problem, depending on how fast the classifier adapts to the current scene. On the one side a general object detector (*e.g.*, [Viola and Jones, 2002b]) is located and on the other side a highly adaptive tracker (*e.g.*, [Grabner and Bischof, 2006]). Our approach is somewhere in between, benefiting from both approaches: (i) be sufficiently adaptive to new appearance and lightning changes, and the simplification of object vs. background and (ii) limit (avoid large) drifting by keeping prior information from the object.

through the exploitation of unlabeled data, we train a supervised learner at the first frame which is never updated throughout the tracking. Roughly speaking, one can think of the prior classifier as a fixed point and the on-line classifier exploring the space around it. This means that the classifier can adapt (or "drift") to new situations but has always the possibility to recover.

9.2.1 Convex Combination of Loss Functions

As proposed above, using on-line SSL for tracking, samples are only passed to the supervised loss at the first frame. During run-time, the unlabeled samples are used to *regularize* the classifier learned at the first frame rather than learning new samples with a powerful supervised loss. As a result, the tracker is more stable and less susceptible to occlusions while it is simultaneously more adaptive than a static classifier [Grabner et al., 2008]. An intuitive explanation for the increased robustness is that a semi-supervised tracker learns

with lower weights in case of reduced confidences of both the current classifier and the prior. However, rapid appearance changes of the target object also result in reduced confidence measurements. In such cases, semi-supervised trackers usually perform inferior to supervised one [Babenko et al., 2009b], where a supervised loss would lead to better results. To alleviate this drawback of SSL, we propose to pass each patch to both a supervised and an unsupervised loss function using the following convex combination

$$\ell(F(\mathbf{x})) = (1-\alpha)\ell_l(F(\mathbf{x})) + \alpha\ell_u(F(\mathbf{x})), \tag{9.2}$$

where ℓ_l is a supervised loss performing self-learning and ℓ_u is a semi-supervised loss where all sub-patches are considered to be unlabeled samples and α steers the importance of the two terms. The supervised loss thus corresponds to self-learning and, as we have argued above, would in principle reduce the stability of the tracker. However, as we showed in Section 5.2, if we make use of more robust loss functions than the common exponential loss, we can handle self-learning and thus the noise that comes along with it up to a certain extend. Hence, we will use SERBoost for the proposed tracker, because it allows for using robust loss functions and α can be set to smaller values than using an exponential loss. This also allows for increasing the influence of the supervised loss in easier scenarios and in cases where the object rapidly changes its appearance, making the SSL tracker more adaptive.

9.2.2 Space-Time Regularization

Digital images provided in form of videos are naturally constrained in form of temporal coherence; *i.e.*, two successive frames are very likely to contain the same content. For object tracking, this means that (i) the background cannot be expected to change significantly between two successive frames and (ii) that the target object does not change its appearance and location significantly from one frame to the other one. This observation is contrast to trackers based on Kalman filters or particle filters, which predict the motion of an object based on previous observations but do not make assumptions about the appearance of an object over space and time [Yilmaz et al., 2006]. In the following we will show that space-time coherence can also be exploited during learning by using the constraint as additional regularization term for an on-line semi-supervised learning algorithm in order to improve the tracking accuracy. Note that Mobahi and Collobert [Mobahi and Collobert, 2009] used a similar principle in order to improve the classification accuracy of deep neural networks from unlabeled videos.

From a semi-supervised learning perspective, the space-time coherence can be easily incorporated into boosting by using manifold regularization techniques. According to these principles, it is enforced that unlabeled samples (in our case patches) that lie spa-

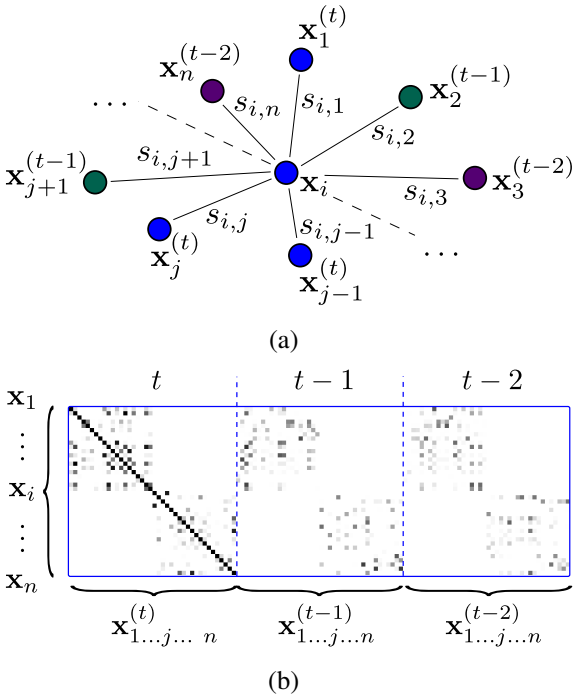


Figure 9.5: Dependency graph between neighboring samples of current and past frames (a) and (b) similarity matrix encoding this relation for 3 frames (block structure due to the grouping of positive and negative samples).

tially close to each other should share the same labels. Extending this idea from space also to time, samples from past frames are also taken into consideration, which is facilitated by the fact that the object motion between frames is limited. However, the influence of frames decreases with a constant factor the more they lie in the past. In the following we will discuss the approach in more detail and show how the space-time constraint can be calculated and how we can transform the thus obtained similarity measure among points into a prior. This prior can then be easily used as a regularization term for semi-supervised boosting.

Similarity Measure

Similarities between samples are described by utilizing a graph based model, which is sketched in Fig. 9.5(a). The similarity of a sample is depending on the spatio-temporal and the visual distance to other samples, with the expectation that close neighbors encode a high similarity.

Spatial-temporal similarity is measured based on the Euclidean distance of samples in

a polar coordinate system with its origin at the current object location:

$$d_S(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\sigma_r}\|r_i - r_j\| + \frac{1}{\sigma_\phi}\|\varphi_i - \varphi_j\|$$
$$d_T(\mathbf{x}_i, \mathbf{x}_j) = |t_i - t_j|. \tag{9.3}$$

r is the distance of a samples from the current object location and φ encodes the angle. Parameter σ_r is chosen in dependence on the object size (*e.g.*, $1/10$ of object size) and parameter σ_ϕ is set such that samples with in an angle (*e.g.*, $\pi/4$) are assumed similar. k is a weighting factor for decreasing influence in time (*e.g.*, $k = -\log(0.8)$)

Visual similarity measures the appearance difference of samples based on their classifier confidences:

$$d_A(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\sigma_A}\left\|F(\mathbf{x}_i) - F(\mathbf{x}_j)\right\|, \tag{9.4}$$

where σ_A controls the deviation for confidence values (*e.g.*, 0.3 if confidences are normalized to lie in $[-1, 1]$)
The resulting similarity measure combines the different distances in a multiplicative way, such that samples are similar only in the case they are close in space, time and appearance:

$$s(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{d_S^2}{2} - \frac{d_A^2}{2} - k d_T\right) \tag{9.5}$$

All similarity measures are combined in a similarity matrix $\mathbf{S}$ with $s_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$.

Prior Calculation

The space-time prior probability $P^+_{p,ST}(\mathbf{x}_i) = P_{p,ST}(y = 1|\mathbf{x}_i)$ of a sample $\mathbf{x}_i$ now depends on its own and its neighboring samples:

$$P^+_{p,ST}(\mathbf{x}_i) = \frac{\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} s(\mathbf{x}_i, \mathbf{x}_j) P^+(\mathbf{x}_j)}{\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} s(\mathbf{x}_i, \mathbf{x}_j)} \tag{9.6}$$

In Eq (9.6) the probability $P^+(\mathbf{x}_j)$ can come from a static or adaptive prior classifier, the current classifier $F(\mathbf{x})$. We apply a simple combination of both. We use a logistic mapping $P^+(\mathbf{x}) = (1 + e^{-2F(\mathbf{x})})^{-1}$ between confidence rated predication and probability. The calculation of prior probabilities is computed efficiently in matrix-vector form

$$\begin{bmatrix} P^+_{p,ST}(\mathbf{x}_1) & \dots & P^+_{p,ST}(\mathbf{x}_n) \end{bmatrix}^{\mathrm{T}} \\ = \mathbf{S} \cdot \begin{bmatrix} \mathbf{p}_{p|t} & \dots & \mathbf{p}_{p|(t-k)} \end{bmatrix}^{\mathrm{T}} \tag{9.7}$$

with $\mathbf{p}_{p|t'} = \begin{bmatrix} P^+_p(\mathbf{x}_1)_{t'} & \dots & P^+_p(\mathbf{x}_n)_{t'} \end{bmatrix}$.

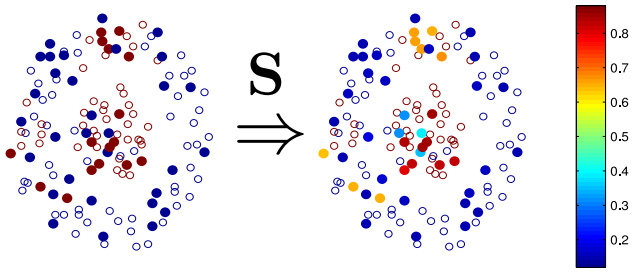


Figure 9.6: Spatial temporal coherence between samples in tracking: Thick points are samples from the current frame, thin circles are samples from previous frames. (left) classification from appearance based prior, (right) the smiliarity encoding $\mathbf{S}$ smoothes of individual (wrong) predictions. Color encoded is the probability of samples belonging to the positive class.

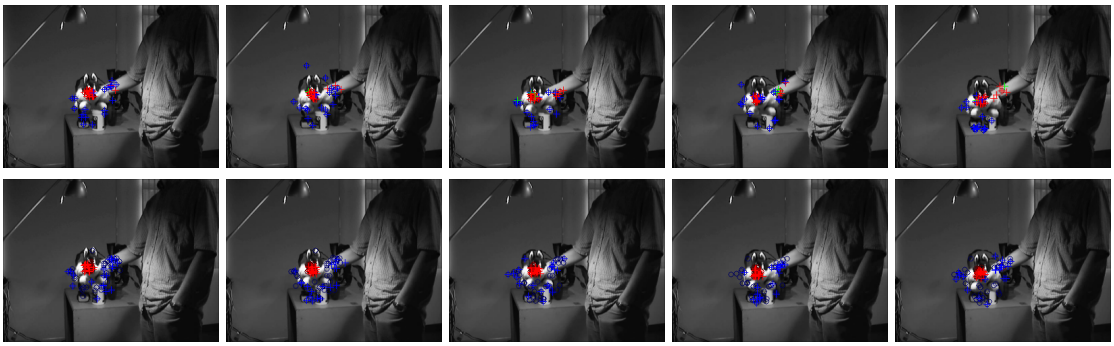
Figure 9.6 demonstrates how the space-time prior calculation influences the algorithm. Standard SERBoost is using the prior information directly, illustrated in the left part. Noise in the prior thus will directly effect the algorithm. By taking spatio-temporal and visual neighbors into consideration noise can be suppressed beforehand.

9.2.3 Tracking and Multiple Instance Learning

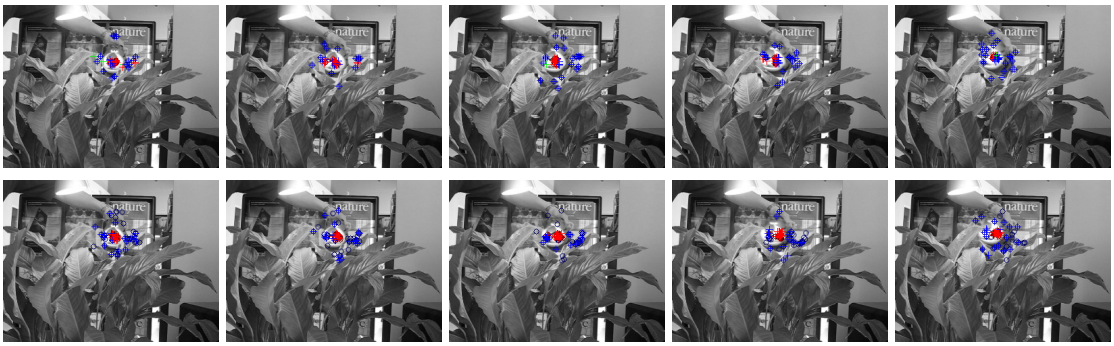
As we have discussed above, one of the main problems facing tracking-by-detection methods is label jitter; *i.e.*, misalignment of object detections. Recently, Babenko *et al.* [Babenko et al., 2009b] proposed an on-line MILBoost formulation and showed that in tracking label jitter can be handled by using multiple instance learning techniques. Using MIL, the classifier in principle is still performing self-learning; however, the allowed positive update area around the current tracker location can be increased and the classifier resolves the ambiguities of where to take the final positive updates by itself, yielding more robust results. See also Figure 9.8 for an illustration. As the MILForest algorithm introduced in Chapter 8 is also on-line learning capable, we can apply it to tracking and will present some experimental results in the following section along with the other methods discussed in this thesis.

9.3 Experiments

The experimental section in principle consists of two parts: In the first part, we will analyze on-line SemiBoost (OSB) and compare to its supervised counterpart where we show that it is less susceptible to drifting. In the second part, we will more quantitatively



(a) *sylv* sequence



(b) *tiger1* sequence

(c) *faceocc1* sequence

Figure 9.7: Comparison between prior calculation from pure prior (top row) and prior smoothing via the additional spatio-temporal constraint (bottom row). Note the smooth and correct prior calculation in comparison due to the incorporation of spatial, temporal and appearance based similarity measures. A green patch means the prior could not decide for one of the two classes.

evaluate all the proposed on-line methods of this work on standard tracking benchmark data sets.

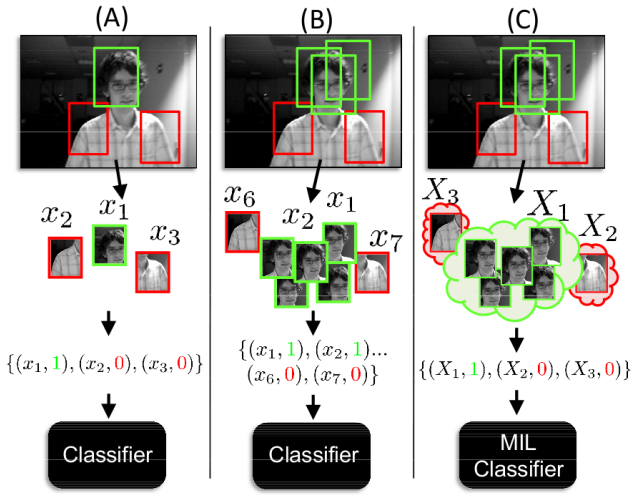


Figure 9.8: Three different ways to update an on-line classifier for tracking (Illustration taken from [Babenko et al., 2009b]): (A) Update with single positive patch (green) and many negative patches [Grabner and Bischof, 2006] (B) Take several positive and negative patches and update a traditional classifier. (C) Put all the putative positive patches into bag and let a multiple instance learner incorporate the positive patches by itself so that it can get the best classification results.

9.3.1 Analysis of On-line SemiBoost

First, we perform experiments demonstrating the specific properties of our tracking approach. Second, we evaluated our tracker on different scenarios showing that we can cope with a large variability of different objects.

Note that the main purpose of the tracking experiments is the comparison of the influence of the different on-line learning methods. Hence, we use simple Haar-like features for representation [Viola and Jones, 2001] which can be calculated efficiently using integral data-structures, did not implement any rotation or scale search and avoid any other engineering methods, although these things would definitely improve the overall results. The performance (speed) depends on the size of the search region which we have defined by enlarging the target region by one third of the object size in each direction (for this region the integral representation is computed). In our experiments we neither use a motion model nor a scaled search window, which both however can be incorporated quite easily. The strong classifier consists of only 25 selectors each with a feature pool of 50 weak classifiers. All experiments are performed on a common 2.4 GHz PC with 2 GB RAM, where we achieve 25 fps tacking speed.

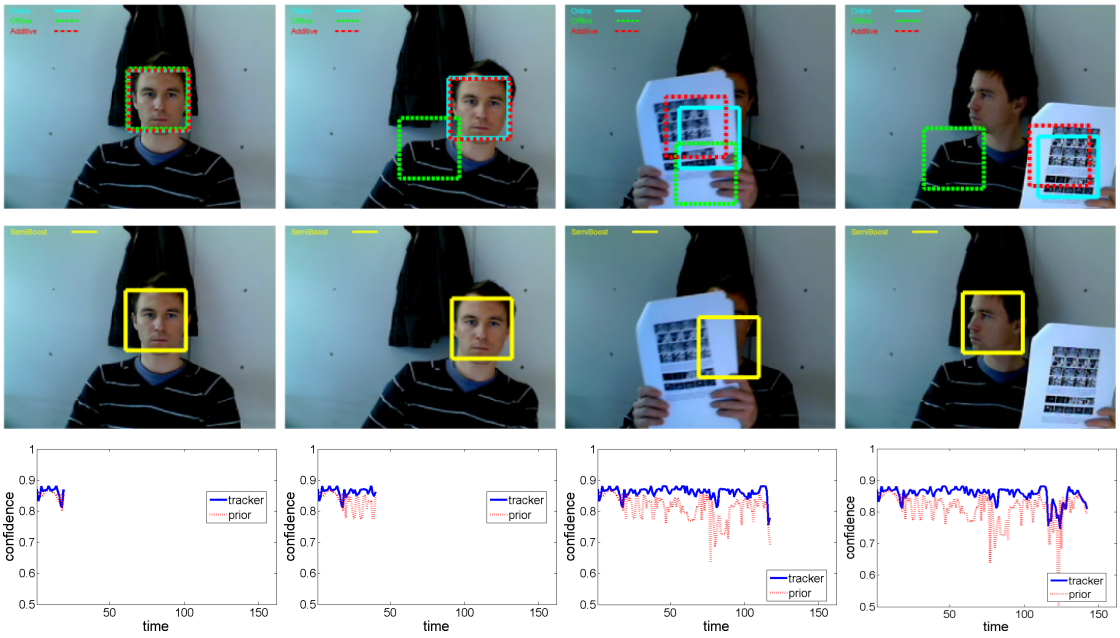


Figure 9.9: Tracking a face in an image sequence under various appearance changes. The first row illustrates three different types of update strategies for the tracker, *i.e.*, (i) on-line boosting (cyan), (ii) prior classifier (red) and (iii) a heuristic combination of (i) and (ii) using the sum-rule, *i.e.*, $0.5(F^P(\mathbf{x}) + F(\mathbf{x}))$ (green). The second row shows the SemiBoost tracker using the same off-line prior. The last row depicts confidence values of the tracked patch over time for the prior and the SemiBoost tracker, respectively.

9.3.2 Illustrations

We illustrate details of our tracker on frontal faces. As prior classifier and for initialization of the tracking process we take the default frontal face detector from OpenCV Version 1.0[1]. This demonstrates that we can use any prior in our method. The primary focus of the experiments is to compare the SemiBoost tracker with other combination methods for the prior and the on-line method[2]. As can be seen from Fig. 9.9, our approach (second row) significantly outperforms the on-line booster, the prior classifier and a heuristic combination of prior and on-line booster (first row). Additionally, even if the prior has very low confidence (third row), the tracker is still able to correctly follow the (side) face. This shows that we can adapt to appearance changes.

Fig. 9.10 depicts some illustrative samples taken for updates for both the on-line tracker and our tracker. As can be observed, while both approaches track the same object, they incorporate totally different updates. After some time the on-line booster performs

[1] `http://sourceforge.net/projects/opencvlibrary/`, 2008/03/16

[2] The OpenCV detector fails on side looking faces.

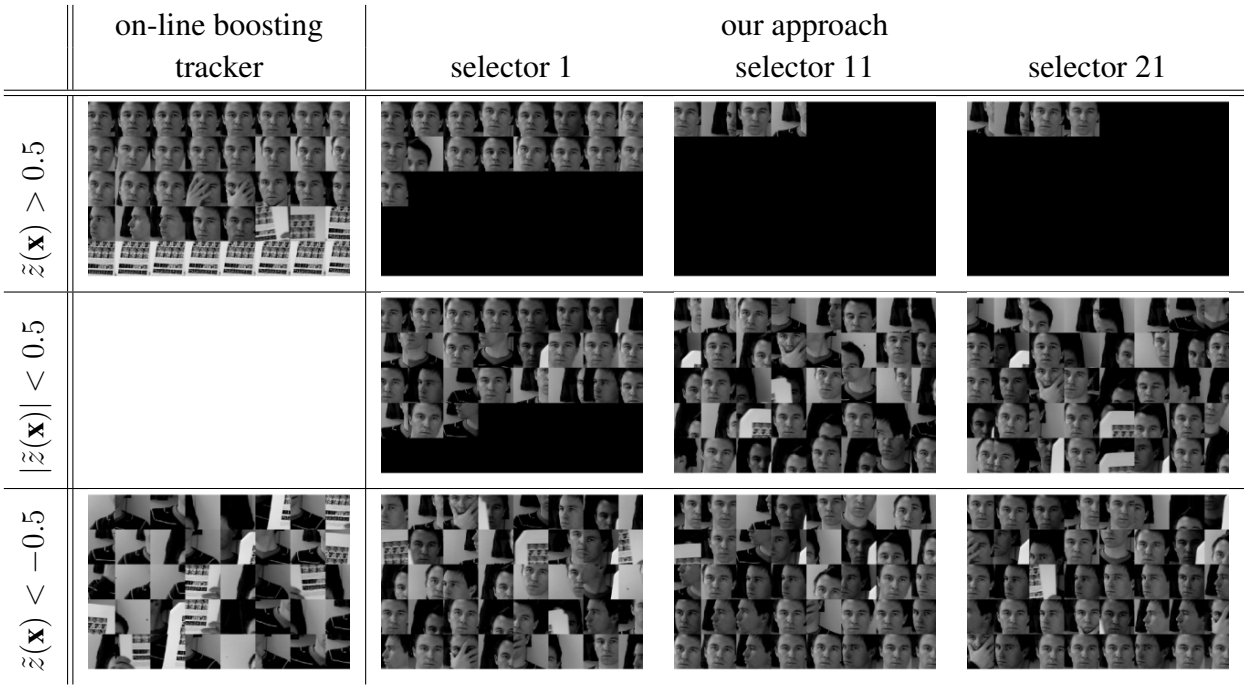
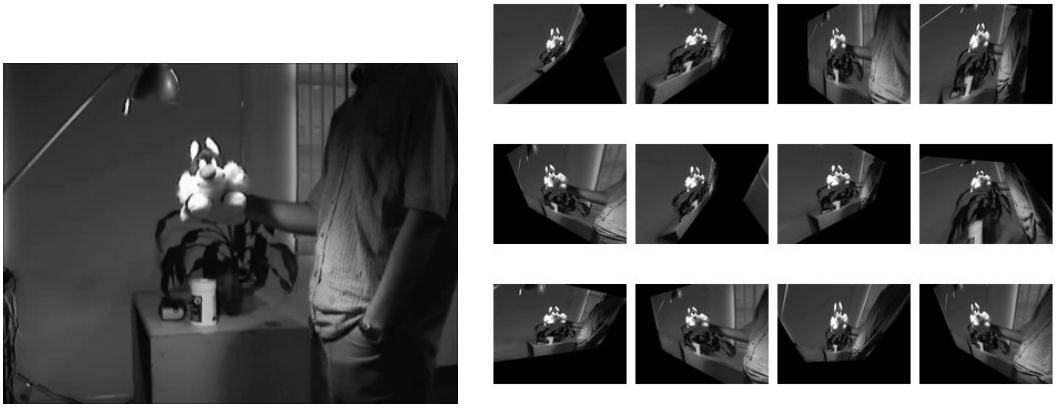


Figure 9.10: Typical updates used for the former on-line boosting tracker (first column). If the tracker loses the object and due to the self-learning update strategy which delivers hard updates, it focuses on another image region. The remaining columns show how samples are incorporated by the SemiBoost tracker. While they are propagated through the selectors, their importance and label can change, with respect to the prior.

wrong updates with still high confidence. This is the main reason for drifting. Furthermore, in the SemiBoost method both sample labels and sample weights change while propagating through the selectors of the SemiBoost tracker while being constant for the former approach. Only such samples are incorporated which are necessary to augment the prior knowledge or invert it in order to be adaptive. Positive samples are inherently treated with caution, *i.e.*, only few positive examples are considered.

9.3.3 One-Shot Prior Learning

For these experiments, the prior is learned from the first frame only. In fact, we build a trainingset $\mathcal{X}_P = \langle \mathbf{x}_o, +1 \rangle \cup \{\langle \mathbf{x}_i, -1 \rangle | \mathbf{x}_i \neq \mathbf{x}_o\}$ where $\mathbf{x}_o$ corresponds to the marked image region and negative samples are generated from the local neighborhood. We added invariance to the training set by creating "virtual" samples [Girosi and Chan, 1995] by performing affine warping on the first frame in order to train a more robust classifier (See also Figure 9.11). Since this trainingset is quite small the time needed to train the prior classifier F^P is negligible. After this one-shot training, the prior classifier is kept constant.



(a) Original Image (b) Virtual Samples

Figure 9.11: Creation of "virtual" samples through affine warping.

In Figure 9.12, we compare our new method to the on-line boosting approach on various tracking scenarios. First, as can be seen in row 1, we are still able to handle challenging appearance changes of the object. Row 2 of Figure 9.12 depicts tracking during a fast movement. Since some incorrect updates and the self-learning strategy of the on-line boosting tracker loses the target and focuses on another part while the semi-supervised tracker is able to re-detect the object. An extremal case is shown in row 3, where we remove the object from the scene. If the object is present again and thanks to the fixed prior our proposed approach has not forgotten the appearance as it is the case for the other tracker and snap to the object again. The next experiment (row 4) focuses on the long term behavior. We chose to track a non-moving object in a static scene for about 1 hour. In order to emphasize the effect we use rather dark illumination conditions. While our proposed tracker stays at the object, the on-line booster starts to drift away. The reason is the accumulation of errors. The final experiment shows a special case of drifting as depicted in the last row of Figure 9.12. Two very similar objects are put together in the scene. Since the pure on-line tracker has not the additional prior information, it is very likely that it is unstable and may switch to another object. Additionally, we choose two public available tracking sequences which have been already used in other publications as can be seen in the last two rows. Our approach performs comparable to the previous on-line tracker on appearance changes (sixth row). After the object was totally occluded (last row), our approach is able to recover the correct object while the former on-line tracker gets confused and starts tracking the second (wrong) car. Additional tracking videos are included as supplementary material.

[2] `http://www.vividevaluation.ri.cmu.edu/datasets/datasets.html`, `2007/06/12`.

9.3.4 Benchmark Sequences

In this part of the experimental section, we will quantitatively evaluate and compare all of the discussed on-line learning variants in this thesis. Therefore, we use eight publicly available sequences including variations in illumination, pose, scale, rotation and appearance, and partial occlusions. The sequences *Sylvester* and *David* are taken from [Ross et al., 2008] and *Face Occlusion 1* is taken from [Adam et al., 2006], respectively. *Face occlusion 2*, *coke*, *Girl*, *Tiger1* and *Tiger2* are taken from [Babenko et al., 2009b]. All video frames are gray scale and of size 320×240.

Evaluation Method

To show the real accuracy of the compared tracking methods, we use the overlap-criterion of the VOC Challenge [Everingham et al., 2007], which is defined as

$$R_T \cap R_{GT} / R_T \cup R_{GT}, \tag{9.8}$$

where R_T is the tracking rectangle and R_{GT} the groundtruth. Since we are interested in the alignment accuracy of our tracker and the tracked object, rather than just computing the raw distance we measure the accuracy of a tracker by computing the average detection score for the entire video. Since it is very difficult or nearly impossible to reach the maximum of 1.0 for this criterion, a value larger than 0.8 can be seen as nearly perfect tracking result, 0.5 to 0.8 would be acceptable.

Tracking performance

As it is the main purpose to compare the learning methods, we use the same simple Haar-like features as representation for all the learning methods, did not implement any rotation or scale search and avoid any other engineering methods. All trackers where initialized by generating 100 virtual samples [Girosi and Chan, 1995] at the first frame using affine transformations. For all boosting methods, we used 50 selectors with each 50 weak classifiers. For all variants of random forests we used 50 trees and a feature pool of randomly selected 500 Haar features.

Influence of the convex combination In the first experiment, we depict the performance of OSERB on four tracking sequences depending on different settings for α. We run each tracker 5 times and report the median with respect to the average overlap score on the whole sequence. The results are given in Figure 9.13 together with an on-line AdaBoost (OAB) performing self-learning and a static classifier only trained at the first frame (PRIOR). All classifiers where trained using additional "virtual" positive samples

generated by applying several affine transformations on the marked target patch of the first frame. As can be seen, the value of α has significant influence on the performance of OSERB. However, it can also be observed that on all sequences for a wide range of α OSERB is able to outperform the competing methods. In particular, OSERB outperforms the competing methods on scenarios where both the target and the background are changing, *e.g.*, *David*, and usually a more adaptive classifier would be expected to perform better. It also outperforms the other methods on sequences where the target object becomes occluded, *e.g.*, *Face occluded 2*, and usually a less adaptive classifier would be preferred. Another surprising result of this experiment is the performance of the non-adaptive tracker which is able to match the performance of OAB on three out of four scenarios and significantly outperforms OAB on the *Girl* sequence.

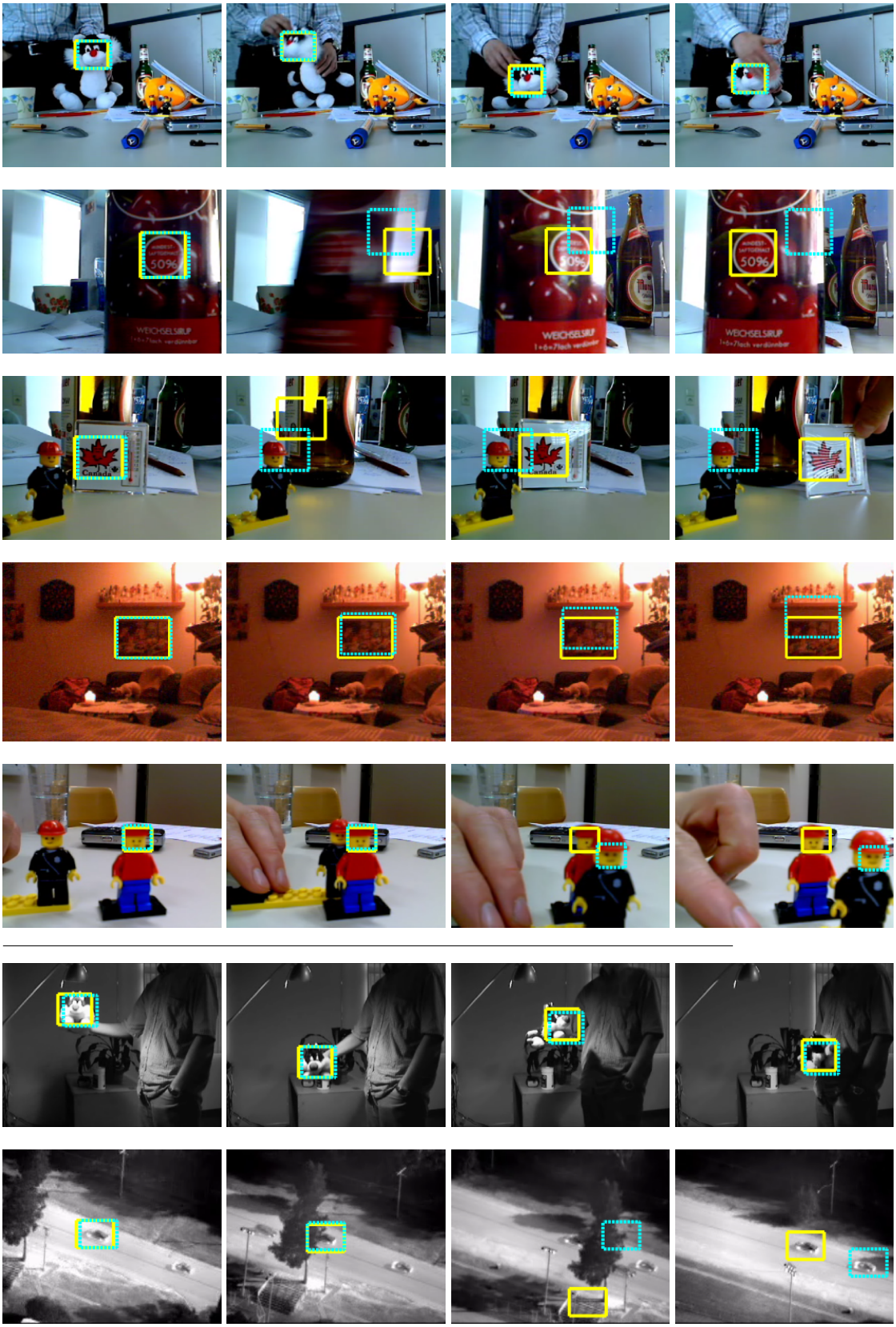
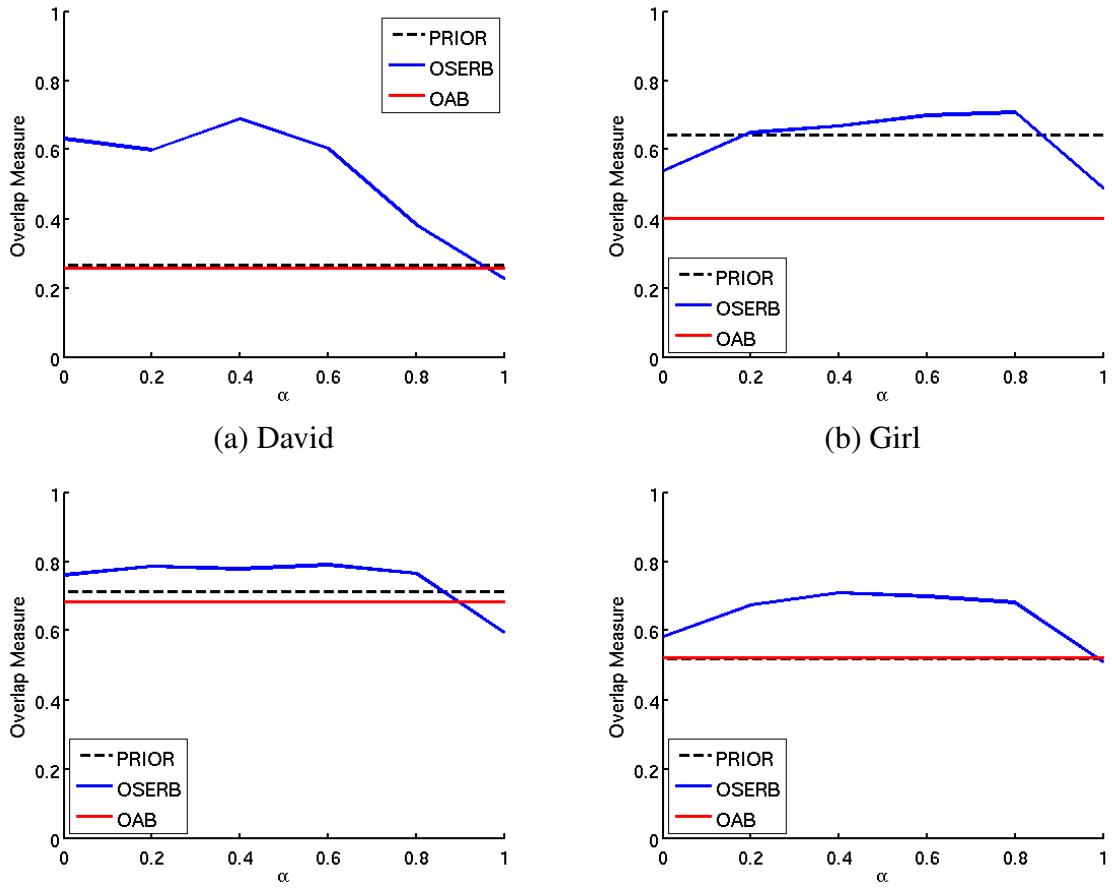


Figure 9.12: Comparisons of the SemiBoost tracker (yellow) and On-line AdaBoost (dotted cyan). SemiBoost is still able to adapt to appearance changes while limiting the drifting. Additionally, results on two public sequences are shown (last two rows). The first sequence have been provided by Lim and Ross ([Ross et al., 2008]) and the second sequence is taken from the VIVID-PETS 2005 dataset.



(a) David (b) Girl

(c) Face occlusion 2 (d) Sylvester

Figure 9.13: Tracking performance depending on different values of α (blue) in comparison to a prior classifier trained at the first frame (black) and and on-line boosting tracking (red).

Quantitative Comparison In Table 9.1 we depict detailed results for all tracking sequences compared to MILBoost [Babenko et al., 2009b], SemiBoost (OSB) and on-line AdaBoost (OAB) [Grabner and Bischof, 2006]. For OSB – as it is not based on robust loss functions – we used only the unsupervised loss by setting $\alpha = 1$. For OSERB, we report the results with $\alpha = 0.5$. Note that for tracking on-line DAS-RF and on-line MILForests are virtual the same; hence, we only report the results for MILForests. STB corresponds to boosting with space-time regularization. PSRF corresponds to on-line semi-supervised RFs using priors. ORF is the on-line random forest method introduced in Chapter 7 and RF is a random forest trained only at the first frame on the target object plus virtual samples. The latter is thus a static tracker similar to the one proposed in [Lepetit and Fua, 2006].

Sequence	OSERB	MILBoost	OSB	OAB	ORF	MILForest	STB	PSRF	RF
sylv	**0.64**	0.61	0.46	0.50	0.53	0.59	0.55	0.58	0.50
david	0.69	0.54	0.31	0.32	0.69	**0.72**	0.66	0.52	0.32
faceocc2	0.77	0.65	0.63	0.64	0.72	0.77	0.66	0.75	**0.79**
tiger1	**0.65**	0.51	0.17	0.27	0.38	0.55	0.43	0.59	0.34
tiger2	0.42	0.50	0.08	0.25	0.43	**0.53**	0.48	0.45	0.32
coke	0.2	0.33	0.08	0.25	0.35	0.35	**0.38**	0.17	0.15
faceocc1	**0.77**	0.63	0.71	0.47	0.71	**0.77**	0.65	0.75	**0.77**
girl	**0.77**	0.53	0.69	0.38	0.70	0.71	0.65	0.63	0.74

Table 9.1: Tracking results on the benchmark sequences measured as average detection window and ground truth overlap over 5 runs per sequence. The best performing method is marked bold-face and the second best underlined, respectively.

As can be seen, for all of the sequences at least one of the proposed methods in this thesis is outperforming the state-of-the-art, *i.e.*, MILBoost or AdaBoost. Additionally, it can be seen that OSERB always performs better than OSB; which speaks for the usage of robust loss functions and the convex combination of both a supervised and an unsupervised loss. The space-time regularized SERBoost (STB) also delivers accurate results; however, it cannot take pace with the SERBoost version that is only regularized with a prior classifier.

We can also see that all random forest-based methods deliver accurate results. As expected, the self-learning variant ORF performs better on sequences that demand an adaptive classifier, such as *tiger1* or *coke*. By contrast, the static random forest classifier RF performs best when it rather comes to occlusions and the adaptivity rate is not so important, as in *faceocc1* and *faceocc2*. The prior regularized random forest version PSRF is somewhere between ORF and RF. On average, the best performing random forest

method is MILForest, which compared to all methods performs best on three sequences and second best on two. Withing the random forest approaches, MILForests seem thus to be a feasible compromise between plasticity and stability.

Taking an overall look at the results, OSERB and MILForest are the two best performing methods. While OSERB has highest accuracy on four out of eight sequences and second highest on two, MILForest performs best on three and second best on two, respectively. Hence, the experiments do not reveal a clear "winning" approach. Concerning the run-time, MILForests learn faster, because they demand less features and benefit from random splitting. However, for evaluation, OSERB is slightly faster because due to the feature selection process, less features have to be evaluated. Note, however, that both approaches run in real-time, *i.e.*, $> 25fps$, on a standard 2.4 GHZ machine with 4 GB memory. Some representative results for OSERB, MILForests and MILBoost are illustrated in Figures 9.14 and 9.14, respectively.

9.4 Summary

In this chapter, we discussed one of the most important applications of on-line boosting, *i.e.*, visual object tracking and showed that the drifting problem can be reduced by formulating this task as one-shot semi-supervised learning using our proposed algorithms. Finally, for the tracking task, we showed that images taken from videos naturally provide spatial and temporal coherence constraints that can be exploited by an on-line semi-supervised, leading to more meaningful regularizations. We compared all of the methods, which where proposed in this thesis, on standard benchmark data sets and showed that both the semi-supervised boosting formulation and the one using random forests are able to outperform the state-of-the-art on this task.

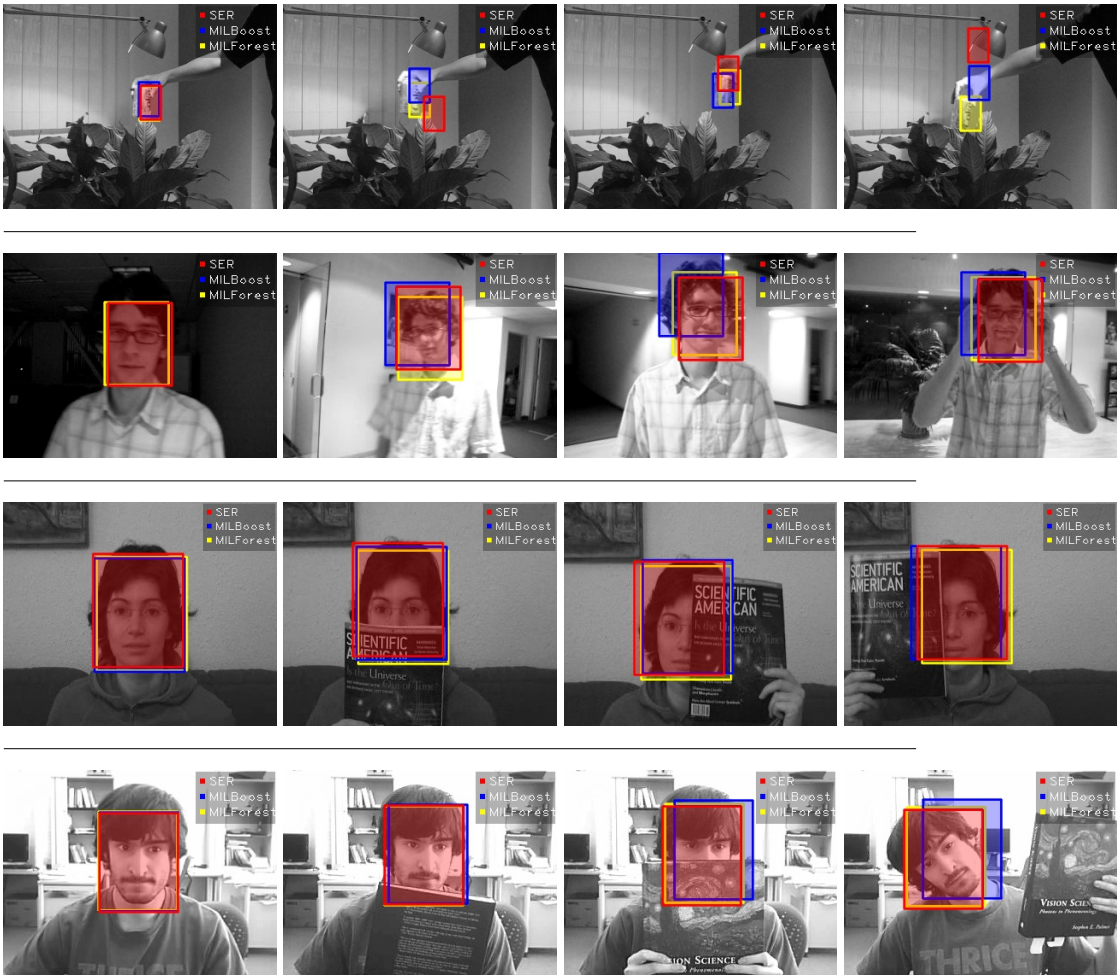


Figure 9.14: Illustrative comparison of SERBoost (red) and MILForests (yellow) and MILBoost (blue) on the *coke*, *david*, *faceocc1* and *faceocc2* sequences.

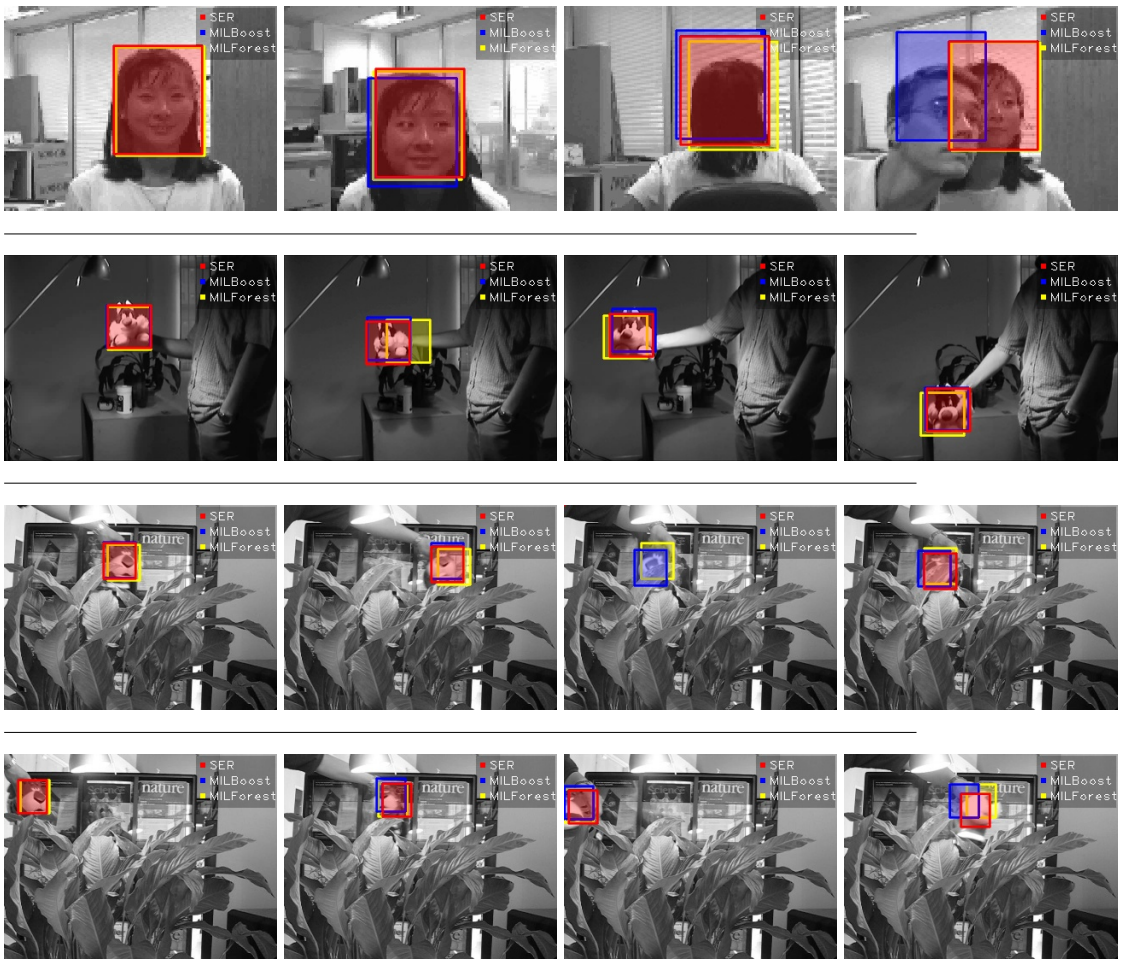


Figure 9.15: Illustrative comparison of SERBoost (red) and MILForests (yellow) and MILBoost (blue) on the *girl*, *sylvester*, *tiger1* and *tiger2* sequences.

Chapter 10

Conclusion

"Learning has just started!", Vladimir Vapnik

In this thesis, we addressed semi-supervised learning algorithms using ensemble methods, and we proposed new SSL methods using boosting and random forests. Our motivation for investigating SSL originated from the fact that machine learning techniques are very important for nowadays computer vision applications but providing huge amounts of human labeled data for those methods is highly impracticable and handicaps their widespread usage. Semi-supervised learning holds the potential to alleviate this problem by enabling learners to exploit large amounts of unlabeled data, thus, reducing the effort to hand-label data to a minimum.

We further concentrated on ensemble methods due to several reasons: First, they are highly effective and have been shown to be competitive alternatives to other methods, for instance, SVMs. Second, they are easy to implement, which guarantees widespread usage as well as easy repeatability and, third, they are already used in many computer vision applications.

We showed that SSL techniques often demand prior knowledge in order to exploit unlabeled data. However, it is often not clear which prior knowledge to take and how to obtain the right priors. In Chapter 4, we introduced a method that combines the usage of visual similarity learning and semi-supervised boosting. The similarities are obtained by learning pair-wise distance functions which has the advantage that (i) the discriminative learned distances best support the discriminative exploitation of the unlabeled data in the further processing and (ii) already a small amount of data is sufficient in order to get proper results. We also showed how to incorporate any source of prior knowledge and how to combine the classifiers if the prior knowledge is provided in form of a classifier. In the experimental section of Chapter 4, we demonstrated several potential applications of this method ranging from classifier improvement to transfer learning in typical multi-camera surveillance settings.

As many applications demand on-line capability, in Chapter 5 we presented a semi-supervised formulation of on-line boosting. We further discussed the lack of robustness of standard boosting in case of label noise and proposed a novel on-line boosting variant using more robust loss functions. Incorporating these insights into semi-supervised boosting further resulted in a new on-line semi-supervised boosting method based on robust logistic loss functions.

The second ensemble method discussed in this thesis are random forests. The motivation for studying RFs stems from the fact that, in recent years, RFs have emerged as a popular classification method, which is increasingly often used in computer vision. However, random forests need large amounts of data in order to enroll their full potential. Therefore, in Chapter 6, we introduced an approach that allows for training random forests with both labeled and unlabeled data, called DAS-RF. We evaluated the method on various machine learning data sets and also for visual object categorization where we showed that we can train a classifier that is inherently able to discriminate between 100 classes on the Caltech-101 data set and also successfully improves its accuracy on unlabeled data. During the learning, semi-supervised RFs treat the hidden class labels of the unlabeled data as additional optimization variables. This means, *e.g.*, that in case of 100 classes, for each sample the classifier has a chance of $\frac{1}{100}$ to estimate the right label; which indicates the complexity of such a task. The method has, furthermore, several appealing characteristics: It is fast, parallel, inherently multi-class and allows for easy detection if unlabeled data either corrupts the results or really helps. Especially the latter characteristics might lead to easier practical usage of the method. Since one may be interested in applying RFs on sequentially arriving data and in dynamic environments, in Chapter 7, we showed how to extend DAS-RFs to on-line learning.

In Chapter 8, we reviewed multiple instance learning. MIL can be applied in numerous computer vision applications, which makes it worthwhile investigating in this machine learning strand. Multiple instance learning is very similar to SSL. In fact, we showed that MIL is a special case of semi-supervised learning, where sub-groups of unlabeled data, *i.e.*, bags, have the additional constraint that at least one instance's real label corresponds to the bag label. Hence, we also introduced a method, MILForests, that extends random forests to multiple instance learning using similar optimization techniques as for DAS-RFs. The method can be used both off-line and on-line. In the experimental section, we demonstrated that MILForests deliver state-of-the-art MIL results but are faster than competitive methods.

Finally, in Chapter 9 we showed that visual object tracking based on on-line classifiers is an *one-shot semi-supervised learning task*, where labeled data is only provided at the first frame of a video sequence and all the remaining frames correspond to unlabeled data. Following this insight, we highlighted that not supervised but semi-supervised learning

methods should be used for tracking, which are inherently designed for exploiting unlabeled data in streaming data scenarios. Thus, using on-line SemiBoost allowed us to change the previously used self-learning tracking loop into a loop where the tracker is "aware" that, beginning with the second frame, secure labeled data cannot be guaranteed and the patches have thus to be incorporated via an unsupervised loss. We demonstrated that the resulting tracker is less susceptible to drifting while being more adaptive than an off-line classifier trained on the first frame. In order to increase the adaptivity we further showed that it is beneficial to during tracking pass sub-patches through both a supervised and an unsupervised loss, however, using robust loss functions which can handle a certain amount of noisy data. We also showed that the multiple-instance formulation of random forests performs better than using the MIL extension for boosting.

10.1 Discussion

This thesis introduced and discussed several semi-supervised learning methods that are applicable in both off-line and on-line scenarios. We showed that unlabeled data, which is available in many vision applications, can be exploited by our methods, and that we can achieve significantly improved results compared to using only labeled data. Both, our semi-supervised versions of boosting and random forests are easy to implement and only comprise small run-time overheads, which allows for straight-forward extension of systems that are already using either of the supervised base-line algorithms.

However, there remain several concerns and open questions that were not discussed throughout this thesis: The semi-supervised boosting methods used in this thesis are based on prior knowledge in form of classifiers. We did not talk about scenarios where we have several classifiers/experts/annotators providing possibly noisy predictions over the unlabeled data. This is often the case in practice, but until now we do not have any mechanism to decide how to benefit from these multiple sources and discard those that are not helping at all. DAS-RFs would in principle allow for testing each prior using the airbag-mechanism. However, considering many priors and large amounts of data it would be impracticable to test each prior first and we rather need a more inherent approach. Furthermore, for boosting, we only discussed binary classification tasks. Recently, however, we showed in [Saffari et al., 2009a] that also for boosting SSL can be done effectively with multiple classes. For further details about multi-class semi-supervised boosting we, henceforth, refer the reader to the thesis of Amir Saffari [Saffari, 2010]

Concerning tracking, we showed that all of our methods, especially, OSERB and MIL-Forests, outperform the state-of-the-art tracking-by-detection methods. Nevertheless, we were not able to indicate a clear winner. Using space-time regularization led to better label predictions of the unlabeled data. Unexpectedly, this does not have a significant

positive effect on the tracking accuracy, although the updates are much nicer. These concerns indicate that currently, from a learning perspective, tracking has arrived at a technically mature state; this means, principle improvements of the learning methods and the regularization approaches do not yield significant improvements for tracking anymore. Thus, research might from now on better focus on investigating better representations and combinations of existing representations in order to gain progress. These considerations also coincide with recent research results [Parikh and Zitnick, 2010] in the area of visual recognition which indicates that nowadays not the learning algorithms but the way how we represent objects is what is lacking behind humans. Nevertheless, as also nowadays representations are usually hand-tuned and it is difficult to design proper representations, one trend is also to learn the representation from a huge set of labeled data [Zeiler et al., 2010, Brown et al., 2010]. Also, one has to question the currently used benchmark data sets and we have probably to come up with new more challenging data sets.

The incorporation of both a supervised and an unsupervised loss during tracking led to higher accuracy of semi-supervised boosting. However, we saw that the values of the steering parameter α have to bet set differently for distinctive sequences in order to yield best results. This is not elegant and arises the question if we can find a control system that during tracking autonomously steers the effect of the supervised and the unsupervised loss, respectively.

10.2 Outlook

The above discussion rises several possibilities and ideas for future research. One of the main concerns is that although there exist already very powerful and highly accurate semi-supervised learning techniques, in practice, people still favor supervised learning and accept hand labeling large amounts of data. So, future research has to definitely focus on making current SSL approaches more suitable for practice and we also have to search for more possible applications. As we shortly discussed in Chapter 3, one problem that most of the existing semi-supervised learning methods have in common and what currently hinders their usage in practice is the fact that they assume both the labeled and the unlabeled data to origin from the same distribution and are hence *i.i.d.*. In practice, this is often not the case; for instance, if someone wants to train a car detector for a specific scene for which he has some labeled data collected and wants to improve the classifier with unlabeled data, however, collected from different scenes. When using currently state-of-the-art SSL methods in such a situation, it is hard to guarantee that the unlabeled data will have any positive effect.

Another practical concern is that although unlabeled data can be easily collected, it is often hard to collect unlabeled data that consist of the right target object. Such a situation,

e.g., may occur if one wants to improve a scene-specific car detector with unlabeled data – probably from different scenarios – that does not contain any cars. For these situations, the question arises if we can design methods that also benefit from unlabeled data that does not contain the target objects. Also falling into this strand of research are recent attempts, *e.g.*, [Lampert et al., 2009, Farhadi et al., 2009, Wang et al., 2010] to describe objects by their attributes and then re-use these attributes to also learn better classifiers from categories where there is only a limited number of samples available.

Future work should probably also concentrate on incorporating special constraints into SSL that are provided by computer vision applications. For instance, as we have seen in the previous chapter, space-time regularization can be such a constraint. Another constraint could be 3-dimensional information or, in general, multi-sensor fusion, *etc.*.

The two main methods we investigated in this thesis were boosting and random forests. Both methods have their individual advantages and disadvantages. Hence, in future work we will concentrate on finding an algorithm that combines the two algorithms.

Appendix A

Publications

My work at the Institute of Computer Graphics and Vision at Graz University of Technology led to the following list of publications. For the sake of completeness, in the following these publications are reported in an inverse chronological order. Note that this thesis is mainly based on a subset of these papers.

(1) Christian Leistner, Amir Saffari, Martin Godec, Bernhard Zeisl and Horst Bischof, *On-line Semi-Supervised Boosting*, submitted to Pattern Recognition, Special Issue on On-line Learning

(2) Christian Leistner, Amir Saffari and Horst Bischof, *MILForests: Multiple Instance Learning with Randomized Trees*, submitted to ECCV 2010

(3) Christian Leistner, Martin Godec, Amir Saffari and Horst Bischof, *On-line Multi-View Forests for Tracking*, submitted to DAGM 2010

(4) Amir Saffari,Christian Leistner and Horst Bischof, *Robust Multi-View Boosting with Priors*, submitted to ECCV 2010

(5) Peter M. Roth, Armin Berger, Christian Leistner and Horst Bischof, *Multiple Instance Learning from Multiple Cameras*, CVPR Workshop on Camera Networks, 2010

(6) Armin Berger, Peter M. Roth, Christian Leistner and Horst Bischof, *Centralized Information Fusion for Learning Object Detectors in Multi-Camera Networks*, OAGM 2010

(7) Martin Godec, Amir Saffari, Christian Leistner and Horst Bischof, *On-line Random Naive Bayes for Tracking*, ICPR 2010

(8) Martin Godec, Christian Leistner, Horst Bischof, Andreas Starzacher and Bernhard Rinner, *Audio Visual Co-Training for Vehicle Classification*, submitted to AVSS 2010

(9) Martin Godec, Christian Leistner, Horst Bischof, Andreas Starzacher and Bernhard Rinner, *Autonomous Audio-Supported Learning of Visual Classifiers for Traffic Monitoring*, IEEE Intelligent Systems, 2010

(10) Amir Saffari,Martin Godec, Thomas Pock, Christian Leistner and Horst Bischof, *On-line Multiclass LPBoost*, In Proc. CVPR, 2010

(11) Jakob Santner, Christian Leistner, Amir Saffari, Thomas Pock and Horst Bischof, *PROST: Parallel Robust Simple Tracking*, In Proc. CVPR, 2010

(12) Bernhard Zeisl, Christian Leistner, Amir Saffari and Horst Bischof, *On-line Semi-Supervised Multiple Instance Boosting*, In Proc. CVPR, 2010

(13) Jakob Santner, Markus Unger, Thomas Pock, Christian Leistner, Amir Saffari and Horst Bischof, *Interactive Texture Segmentation using Random Forests and Total Variation*, In Proc. BMVC, 2009

(14) Peter M. Roth, Christian Leistner, Helut Grabner and Horst Bischof, *On-line Learning in Multiple Camera Networks*, In Multicamera Networks, Principles and Applications, Academic Press, 2009

(15) Martin Godec, Helmut Grabner, Christian Leistner and Horst Bischof, *Speeding up Semi-Supervised On-line Boosting for Tracking*, AAPR / ÖAGM, 2009

(16) Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec and Horst Bischof, *On-line Random Forests*, 3re IEEE ICCV Workshop on OLCV, 2009

(17) Christian Leistner, Amir Saffari and Horst Bischof, *On Robustness of On-line Boosting: A Competitive Study*, 3re IEEE ICCV Workshop on OLCV, 2009

(18) Christian Leistner, Amir Saffari, Jakob Santer and Horst Bischof, *Semi-supervised Random Forests*, ICCV, 2009

(19) Amir Saffari, Christian Leistner and Horst Bischof, *Regularized Multiclass Semi-supervised Boosting*, CVPR, 2009

(20) Christian Leistner, Peter M. Roth and Horst Bischof, *Visual On-line Learning in Distributed Camera Networks*, In Proc. IEEE ICDSC, 2008

(21) Peter M. Roth, Helmut Grabner, Christian Leistner, Martin Winter and Horst Bischof, *Interactive Learning a Person Detector: Fewer Clicks, less Frustration*, In Proc. AAPR / ÖAGM, 2008

(22) Christian Leistner, Helmut Grabner and Horst Bischof, *Semi-supervised Boosting using Visual Similarity Learning*, In Proc. IEEE CVPR, 2008

(23) Helmut Grabner, Christian Leistner and Horst Bischof, *Semi-supervised On-line Boosting for Robust Tracking*, In Proc. ECCV, 2008 T

(24) Helmut Grabner, Christian Leistner and Horst Bischof, *Time-Dependent On-line Boosting for Robust Background Modeling*, In Proc. VISAP, 2008

(25) Clemens Arth, Christian Leistner and Horst Bischof, *Object Reacquisition and Tracking in Large-Scale Smart Camera Networks*, In Proc. IEEE ICDSC, 2008

(26) Clemens Arth, Christian Leistner and Horst Bischof, *Robust Local Features and Their Application in Self-Calibration and Object Recognition on Embedded Systems*, In Proc. IEEE CVPR Workshop on ECV, 2007

(27) Clemens Arth, Christian Leistner and Horst Bischof, *TRICAM: An Embedded Platform for Remote Traffic Surveillance*, In Proc. IEEE CVPR Workshop on ECV, 2006

Appendix B

Acronyms

DA Deterministic Annealing

EM Expectation Maximization

GE Generalization Error

HOG Histograms of Oriented Gradients

KL Kullback-Leibler

MIL Multiple Instance Learning

MLE Maximum Likelihood Estimate

OAB On-line AdaBoost

OOBE Out-of-bag Error

ORF On-line Random Forest

OSB On-line SemiBoost

OSERB On-line SERBoost

PSRF Prior-regularized Semi-supervised Random Forests

RBF Radial Basis Function

RF Random Forest

SSL Semi-supervised Learning

STB Space-time Boost

SVM Support Vector Machine

XR Expectation Regularization

Bibliography

[Abney, 2007] Abney, S. (2007). *Semi-Supervised Learning for Computational Linguistics*. Computer Science and Data Analysis. Chapman & Hall/CRC, first edition.

[Adam et al., 2006] Adam, A., Rivlin, E., and Shimshoni, I. (2006). Robust fragments-based tracking using the integral histogram. In *CVPR*.

[Alpaydin, 2010] Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2 edition.

[Andrews et al., 2003] Andrews, S., Tsochandaridis, I., and Hofman, T. (2003). Support vector machines for multiple-instance learning. *Adv. in Neural. Inf. Process. Syst.*, 15:561–568.

[Avidan, 2007] Avidan, S. (2007). Ensemble tracking. *PAMI*, 29(2):261–271.

[Babenko et al., 2009a] Babenko, B., Branson, S., and Belongie, S. (2009a). Similarity metrics for categorization: from monolithic to category specific. In *ICCV*.

[Babenko et al., 2009b] Babenko, B., Yang, M.-H., and Belongie, S. (2009b). Visual tracking with online multiple instance learning. In *CVPR*.

[Bahrick et al., 2002] Bahrick, L. E., Lickliter, R., and Flom, R. (2002). Intersensory redundancy guides the development of selective attention, perception and cognition in infancy. *Intersensory Redundancy Guides the Development of Selective Attention, Perception, and Cognition in Infancy*, 13(3):99–102.

[Baker and Matthews, 2004] Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *IJCV*, pages 221–255.

[Balcan et al., 2004] Balcan, M.-F., A.F, and Yang, K. (2004). Co-training and expansion: Towards bridging theory and practice. In *Advances in Neural Information Processing Systems*. MIT Press.

[Balcan and Blum, 2005] Balcan, M.-f. and Blum, A. (2005). A pac-style model for learning from labeled and unlabeled data. In *In Proceedings of the 18th Annual Conference on Computational Learning Theory (COLT*, pages 111–126.

[Bar-Shalom, 2008] Bar-Shalom, Y. (2008). Tracking and data association.

[Basak, 2004] Basak, J. (2004). Online adaptive decision trees: Pattern classification and function approximation. *Neural Comput.*, 18:2062–2101.

[Belkin et al., 2006] Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. 7:2399–2434.

[Ben-David et al., 2008] Ben-David, S., Lu, T., and Pal, D. (2008). Does unlabeled data provably help?

[Bengio et al., 2009] Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *ICML*.

[Bennett and Demiriz, 1999] Bennett, K. and Demiriz, A. (1999). Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, volume 11, pages 368–374.

[Bennett et al., 2002] Bennett, K., Demiriz, A., and Maclin, R. (2002). Exploiting unlabeled data in ensemble methods.

[Berger et al., 1996] Berger, A. L., Della Pietra, V. J., and Della Pietra, S. A. (1996). A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.

[Blockeel et al., 2005] Blockeel, H., Page, D., and Srinivasan, A. (2005). Multi-instance tree learning. In *ICML*.

[Blum, 1996] Blum, A. (1996). On-line algorithms in machine learning. In *Online Algorithms*, pages 306–325.

[Blum and Chawla, 2000] Blum, A. and Chawla, S. (2000). Learning from labeled and unlabeled data using graph mincuts. In *ICML*.

[Blum and Kalai, 1998] Blum, A. and Kalai, A. (1998). A note on learning from multiple instance examples. pages 23–29.

[Blum et al., 2001] Blum, A., Lafferty, J., Rwebangria, M. R., and Reddy, R. (2001). Semi-supervised learning using randomized mincuts. In *ICML*.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. pages 92–100.

[Bordes et al., 2007] Bordes, A., Bottou, L., Gallinari, P., and Weston, J. (2007). Solving multiclass support vector machines with larank. In *ICML*.

[Bosch et al., 2007] Bosch, A., Zisserman, A., and Munoz, X. (2007). Image classification using random forests and ferns. In *ICCV*.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[Breiman, 1996a] Breiman, L. (1996a). Baggig predictors. *Machine Learning*, 24(4):123–140.

[Breiman, 1996b] Breiman, L. (1996b). Out-of-bag estimates. Technical report.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*.

[Brown et al., 2010] Brown, M., Hua, G., and Winder, S. (2010). Discriminant learning of local image descriptors. *PAMI*.

[Bunescu and Mooney, 2007] Bunescu, R. and Mooney, R. (2007). Multiple instance learning for sparse positive bags. In *ICML*.

[Caruana et al., 2008] Caruana, R., Karampatziakis, N., and Yessenalina, A. (2008). An empirical evaluation of supervised learning in high dimensions.

[Chang and Lin, 2001] Chang, C.-C. and Lin, C.-J. (2001). Libsvm: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

[Chapelle et al., 2006] Chapelle, O., Schoelkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. Adaptive Computation and Machine Learning. The MIT Press.

[Chapelle et al., 2003] Chapelle, O., Weston, J., and Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. In *NIPS*, volume 15 of *NIPS*, pages 585–592.

[Chapelle and Zien, 2005] Chapelle, O. and Zien, A. (2005). Semi-supervised classification by low density separation. In *AISTATS*.

[Chen and Wang, 2008] Chen, K. and Wang, S. (2008). Regualarized boost for semi-supervised learning.

[Chen et al., 2006] Chen, Y., Bi, J., and Wang, J. (2006). Miles: Multiple-instance learning via embedded instance selection. In *IEEE PAMI*.

[Christoudias et al., 2008a] Christoudias, C. M., Urtasun, R., and Darrell, T. (2008a). Multi-view learning in the presence of view disagreement. In *In Proc. of the Conference on Uncertainty in Artifcial Intelligence*.

[Christoudias et al., 2008b] Christoudias, C. M., Urtasun, R., Kapoor, A., and Darrell, T. (2008b). Co-training with noisy perceptual observations. In *CVPR*.

[Collins and Singer, 1999] Collins, M. and Singer, Y. (1999). Unsupervised models for the names entity classification.

[Comaniciu et al., 2003] Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. 25(5).

[Dai et al., 2009] Dai, W., Jin, O., Xue, G.-R., Yang, Q., and Yu, Y. (2009). Eigentransfer: A unified framework for transfer learning. In *ICML*.

[d'Alche Buc et al., 2002] d'Alche Buc, F., Grandvalet, Y., and Ambroise, C. (2002). Semi-supervised marginboost. In *Advances in Neural Information Processing Systems*. MIT Press.

[Dasgupta et al., 2002] Dasgupta, S., Littman, M. L., and McAllester, D. (2002). Pac generalization bounds for co-training. *Advances in Neural Information Processing Systems*, 14:375 – 382.

[Dietterich, 1998] Dietterich, T. (1998). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157.

[Dietterich et al., 1997] Dietterich, T., Lathrop, R., and Lozano-Perez, T. (1997). Solving the multiple-instance problem with axis-paralle rectangles. In *Artifical Intelligence*.

[Domingo and Watanabe, 2000] Domingo, C. and Watanabe, O. (2000). MadaBoost: A modification of AdaBoost. In *Proc. COLT*, pages 180–189.

[Duda et al., 2001] Duda, R., Hart, P., and Stork, D. (2001). *Pattern Classification*. Wiley-Interscience, 2 edition.

[Everingham et al., 2007] Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The pascal visual object class challenge 2007. In *VOC*.

[Farhadi et al., 2009] Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. (2009). Describing objects by attributes. In *CVPR*.

[Fergus et al., 2009] Fergus, R., Torralba, A., and Weiss, Y. (2009). Semi-supervised learning in gigantice image collections. In *NIPS*.

[Foulds and Frank, 2008] Foulds, J. and Frank, E. (2008). Revisiting multi-instance learning via embedded instance selection.

[Freund, 1995] Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285.

[Freund, 2000] Freund, Y. (2000). An adaptive version of the boost by majority algorithm. In *Proc. COLT*, volume 43.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proc. ICML*, pages 148–156.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

[Freund and Schapire, 1999] Freund, Y. and Schapire, R. (1999). A short introduction to boosting. *Journal of the Japanese Society for Artificial Intelligence*, 14(5):771–780.

[Friedman, 2001] Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232.

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibishirani, R. (2001). *The elements of Statistical Learning*. springer, second edition.

[Friedman et al., 2000] Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407.

[Frome et al., 2007] Frome, A., Singer, Y., Sha, F., and Malik, J. (2007). Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*.

[Gall and Lempinsky, 2009] Gall, J. and Lempinsky, V. (2009). Class-specific hough forests for object detection. In *CVPR*.

[Gehler and Chapelle, 2007] Gehler, P. and Chapelle, O. (2007). Deterministic annealing for multiple-instance learning. In *AISTATS*.

[Geurts et al., 2006] Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees.

[Girosi and Chan, 1995] Girosi, F. and Chan, N. T. (1995). Prior knowledge and the creation of "virtual" examples for rbf networks. In *In Neural networks for signal processing, Proceedings of the 1995 IEEE-SP Workshop*, pages 201–210.

[Goldberg et al., 2009] Goldberg, A., Zhu, X., Singh, A., Xu, Z., and Nowak, R. (2009). Multi-manifold semi-supervised learning.

[Goldberg et al., 2008] Goldberg, A. B., Li, M., and Zhu, X. (2008). Online manifold regularization: A new learning setting and empirical study. In *ECMLPKDD*.

[Grabner and Bischof, 2006] Grabner, H. and Bischof, H. (2006). On-line boosting and vision. In *CVPR*, volume 1, pages 260–267.

[Grabner et al., 2008] Grabner, H., Leistner, C., and Bischof, H. (2008). On-line semi-supervised boosting for robust tracking. In *ECCV*.

[Grossberg, 1998] Grossberg, S. (1998). Competitive learning: From interactive activation to adaptive resonance. *Neural networks and natural intelligence*, pages 213–250.

[Guillaumin and Schmid, 2010] Guillaumin, M. and Schmid, J. V. C. (2010). Multimodal semi-supervised learning for image classification. In *CVPR*.

[Hadamard, 1902] Hadamard, J. (1902). Sur les problémes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, 13.

[Hertz et al., 2004] Hertz, T., Bar-Hillel, A., and Weinshall, D. (2004). Learning distance functions for image retrieval. In *CVPR*, volume 2, pages 570–577.

[Hinton and Sejnowski, 1999] Hinton, G. and Sejnowski, T. J. (1999). *Unsupervised Learning: Foundations of Neural Computation*. MIT Press, first edition.

[Huang et al., 2006] Huang, T.-M., Kecman, V., and Kopriva, I. (2006). *Kernel Based Algorithms for Mining Huge Data Sets*. Springer, 1 edition.

[Jain et al., 2008] Jain, P., Kulis, B., and Grauman, K. (2008). Online metric learning and fast similarity search. In *NIPS*.

[Javed et al., 2005] Javed, O., Ali, S., and Shah, M. (2005). Online detection and classification of moving objects using progressively improving detectors. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume I, pages 696–701.

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209.

[Kääriäinen, 2005] Kääriäinen, M. (2005). Generalization error bounds using unlabeled data.

[Kalman, 2008] Kalman, R. E. (2008). A new approach to linear filtering and prediction problems.

[Kearns and Valiant, 1994] Kearns, M. and Valiant, L. G. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the Association for Computing Machinery*, 71(1):67–95.

[Keeler et al., 1990] Keeler, J., Rumelhart, D., and Leow, W. (1990). Integrated segmentation and recognition of hand-printed numerals. In *NIPS*.

[Kegel and Wang, 2005] Kegel, B. and Wang, L. (2005). Boosting on manifolds: Adaptive regularization of base classifiers. In *Advances in Neural Information Processing Systems*.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

[Lafferty and Wasserman, 2008] Lafferty, J. and Wasserman, L. (2008). Statistical analysis of semi-supervised regression. In *Advances in Neural Information Processing Systems*.

[Lampert et al., 2009] Lampert, C., Nickisch, H., and Harmeling, S. (2009). Learning to detect object classes by between-class attribute transfer. In *CVPR*.

[Leistner et al., 2008] Leistner, C., Grabner, H., and Bischof, H. (2008). Semi-supervised boosting using visual similarity learning. In *CVPR*.

[Leistner et al., 2009a] Leistner, C., Saffari, A., Roth, P. M., and Bischof, H. (2009a). On robustness of on-line boosting - a competitive study. In *OLCV (ICCV)*.

[Leistner et al., 2009b] Leistner, C., Saffari, A., Santner, J., and Bischof, H. (2009b). Semi-supervised random forests. In *ICCV*.

[Lepetit and Fua, 2006] Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. In *CVPR*.

[Leskes and Torenvliet, 2008] Leskes, B. and Torenvliet, L. (2008). The value of agreement a new boosting algorithm. *Journal of Computer and System Sciences*.

[Lettvin et al., 1959] Lettvin, J. Y., Maturana, H. R., McCulloch, W., and Pitts, W. (1959). What the frog's eye tells the frog's brain. In *Inst. Radio Engr.*, volume 47, pages 1940–1951.

[Levin et al., 2003] Levin, A., Viola, P., and Freund, Y. (2003). Unsupervised improvement of visual detectors using co-training. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume I, pages 626–633.

[Li et al., 2007] Li, Y., Ai, H., Yamashita, T., Lao, S., and Kawade, M. (2007). Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans. In *CVPR*, pages 1–8.

[Lin, 2004] Lin, Y. (2004). A note on margin-based loss functions in classification. *Statistics & Probability Letters*, 68(1):73–82.

[Lin and Jeon, 2006] Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest nieghbors. *Journal of the American Statistical Association*.

[Liu and Leordeanu, 2005] Liu, M. C. Y. and Leordeanu, M. (2005). Online selection of disriminative tracking features. In *PAMI*.

[Liu et al., 2009] Liu, R., Cheng, J., and Lu, H. (2009). A robust boosting tracker with minimum error bound in a co-training framework. In *ICCV*.

[Livingstone, 2008] Livingstone, M. (2008). *Vision and Art: The Biology of Seeing*. Abrams, New York, first edition.

[Loeff et al., 2008] Loeff, N., Forsyth, D., and Ramachandran, D. (2008). Manifoldboost: Stagewise function approximation for fully, semi- and un-supervised learning. In *ICML*.

[Long and Servedio, 2005] Long, P. M. and Servedio, R. A. (2005). Martingale boosting. In *COLT*.

[Long and Servedio, 2008a] Long, P. M. and Servedio, R. A. (2008a). Adaptive martingale boosting. In *NIPS*.

[Long and Servedio, 2008b] Long, P. M. and Servedio, R. A. (2008b). Random classification noise defeats all convex potential boosters. In *ICML*.

[Maclin and Opitz, 1997] Maclin, R. and Opitz, D. (1997). An empirical evaluation of bagging and boosting. In *Proc. National Conf. on Artificial Intelligence*, pages 546–551.

[Mallapragada et al., 2009] Mallapragada, K., Jin, R., Jain, A. K., and Liu, Y. (2009). Semiboost: Boosting for semi-supervised learning. *PAMI*.

[Mangasarian and Wild, 2005] Mangasarian, O. and Wild, E. (2005). Multiple-instance learning via successive linear programming. Technical report.

[Mann and Mccallum, 2007] Mann, G. S. and Mccallum, A. (2007). Simple, robust, scalable semi-supervised learning via expectation regularization. In *ICML*, pages 593–600.

[Maron and Lozano-Perez, 1997] Maron, O. and Lozano-Perez, T. (1997). A framework for multiple-instance learning. In *NIPS*.

[Marr, 1982] Marr, D. (1982). *Vision: A computational investigation into the human representation and Processing of Visual Information*. W. H. Freeman, first edition.

[Masnadi-Shirazi and Vasconcelos, 2008] Masnadi-Shirazi, H. and Vasconcelos, N. (2008). On the design of loss functions for classification: Theory, robustness, and savageboost. In *Advances in Neural Information Processing Systems*.

[Mason et al., 1999] Mason, L., Baxter, J., and Frean, P. (1999). Boosting algorithms as gradient descent. *Advances in Large Margin Classifiers*, pages 221–247.

[Matthews et al., 2004] Matthews, I., Ishikawa, T., and Baker, S. (2004). The template update problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26:810 – 815.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1 edition.

[Mobahi and Collobert, 2009] Mobahi, H. and Collobert, R. (2009). Deep learning from temporal coherence in video. In *ICML*.

[Moosmann et al., 2006] Moosmann, F., Triggs, B., and Jurie, F. (2006). Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, pages 985–992.

[Nigam et al., 2006] Nigam, K., McCallum, A., and Mitchell, T. (2006). Semi-supervised text classification using em. *Semi-Supervised Learning,MIT Press*, pages 33–55.

[Nowak and Jurie, 2007] Nowak, E. and Jurie, F. (2007). Learning visual similarity measures for comparing never seen objects. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1–8.

[Olshausen and Field, 1996] Olshausen, B. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609.

[Oza, 2001] Oza, N. (2001). *Online Ensemble Learning*. PhD thesis, University of California, Berkeley.

[Oza and Russell, 2001] Oza, N. and Russell, S. (2001). Online bagging and boosting. In *Proceedings Artificial Intelligence and Statistics*, pages 105–112.

[Özuysal et al., 2007] Özuysal, M., Fua, P., and Lepetit, V. (2007). Fast keypoint recognition in ten lines of code. In *CVPR*.

[Pakkanen et al., 2004] Pakkanen, J., Iivarinen, J., and Oja, E. (2004). The evolving tree—a novel self-organizing network for data analysis. *Neural Process. Lett.*, 20(3):199–211.

[Palmer, 1999] Palmer, S. E. (1999). *Vision Science: Photons to Phenomenology*. The MIT Press, first edition.

[Papageorgiou et al., 1998] Papageorgiou, C., Oren, M., and Poggio, T. (1998). A general framework for object detection. In *Proc. IEEE Intern. Conf. on Computer Vision*, pages 555–562.

[Parikh and Zitnick, 2010] Parikh, D. and Zitnick, C. L. (2010). The role of features, algorithms and data in visual recognition. In *CVPR*.

[Pfahringer et al., 2007] Pfahringer, B., Holmes, G., and Kirkby, R. (2007). New options for hoeffding trees. *Advances in Artificial Intelligence*, 71.

[Prinzie and den Poel, 2007] Prinzie, A. and den Poel, D. V. (2007). Random multiclass classification: Generalizing random forests to random mnl and nb. In *Database and Expert Systems Applications*.

[Raina et al., 2007] Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. In *ICML*.

[Rigollet, 2007] Rigollet, P. (2007). Generalization error bounds in semi-supervised classification under the cluster assumption. *JMLR*, 8:1369–1392.

[Rose, 1998] Rose, K. (1998). Deterministic annealing, constrained clustering, and optimization. In *IJCNN*.

[Ross et al., 2008] Ross, D., Lim, J., Lin, R.-S., and Yang, M.-H. (2008). Incremental learning for robust visual tracking. *IJCV*.

[Rudin et al., 1999] Rudin, C., Daubechies, I., and Schapire, R. E. (1999). The dynamics of adaboost: Cyclic behavior amd convergence of margins. *JMLR*, 14(5):771–780.

[Ruffo, 2000] Ruffo, G. (2000). *Learning single and multiple instance decision trees for computer security applications*. PhD thesis.

[Saffari, 2010] Saffari, A. (2010). *Boosting for Multi-class Semi-supervised Learning*. PhD thesis.

[Saffari et al., 2008] Saffari, A., Grabner, H., and Bischof, H. (2008). Serboost: Semi-supervised boosting with expectation regularization. In *ECCV*, pages 588–601.

[Saffari et al., 2009a] Saffari, A., Leistner, C., and Bischof, H. (2009a). Regularized multi-class semi-supervised boosting. In *CVPR*.

[Saffari et al., 2009b] Saffari, A., Leistner, C., Godec, M., Santner, J., and Bischof, H. (2009b). On-line random forests. In *OLCV*.

[Schapire et al., 2002] Schapire, R. E., M.Rochery, Rahim, M., and Gupta, N. (2002). Incorporating prior knowledge into boosting. In *ICML*.

[Schoelkopf and Smola, 2002] Schoelkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, first edition.

[Shakhnarovich, 2005] Shakhnarovich, G. (2005). Learning task-specific similarity. *Massachusetts Institute of Technology*.

[Shakhnarovich et al., 2005] Shakhnarovich, G., Darrell, T., and Indyk, P. (2005). *Nearest-Neighbor Methods in Learning and Vision*. MIT Press.

[Shalev-Shwartz, 2007] Shalev-Shwartz, S. (2007). *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University of Jerusalem.

[Sharp, 2008] Sharp, T. (2008). Implementing decision trees and forests on a gpu. In *ECCV*.

[Shen et al., 2005] Shen, D., Zhang, J., Su, J., Zhou, G., and Tan, C.-L. (2005). A collaborative ability measurement for co-training. *Springer*.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *CVPR*.

[Shotton et al., 2008] Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image catergorization and segmentation. In *CVPR*.

[Sindhwani et al., 2006] Sindhwani, V., Keerthi, S. S., and Chapelle, O. (2006). Deterministic annealing for semi-supervised kernel machines. In *ICML*.

[Singh et al., 2009] Singh, A., Nowak, R., and Zhu, X. (2009). Unlabeled data: Now it helps, now it doesn't. In *Advances in Neural Information Processing Systems*.

[Socher and Fei-Fei, 2010] Socher, R. and Fei-Fei, L. (2010). Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *CVPR*, San Francisco, CA.

[Stikic and Schiele, 2009] Stikic, M. and Schiele, B. (2009). Activity recognition from sparsely labeled data using multi-instance learning. In *Proceedings of the 4th International Symposium on Location and Context Awareness (LoCA'09)*, page 156–173, Tokyo, Japan. Springer, Springer.

[Szeliski, 2010] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer, 1 edition.

[Tang et al., 2007] Tang, F., Brennan, S., Zao, Q., and Tao, W. (2007). Co-tracking using semi-supervised support vector machines. In *ICCV*.

[Tieu and Viola, 2000] Tieu, K. and Viola, P. (2000). Boosting image retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 228–235.

[Torralba et al., 2008] Torralba, A., Fergus, R., and Weiss, Y. (2008). Small codes and large databases for recognition. In *CVPR*.

[Utgoff et al., 1997] Utgoff, E., Bergman, N., and Clouse, J. (1997). Decision tree induction based on efficient tree restructuring. *Machine Learning*.

[Vandist et al., 2009] Vandist, K., Schryver, M., and Rosseel, Y. (2009). Semisupervised category learning: The impact of feedback in learning the information-integration task. *Attention,Perception, & Psychophysics*, 71(2):328 – 341.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

[Vijayanarasimhan S. Grauman, 2008] Vijayanarasimhan S. Grauman, K. (2008). Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization. In *CVPR*.

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume I, pages 511–518.

[Viola and Jones, 2002a] Viola, P. and Jones, M. (2002a). Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing Systems*. MIT.

[Viola and Jones, 2002b] Viola, P. and Jones, M. (2002b). Robust real-time object detection. *Intern. Journal of Computer Vision*.

[Viola et al., 2006] Viola, P., Platt, J., and Zhang, C. (2006). Multiple instance boosting for object detection. In *NIPS*.

[Wang et al., 2010] Wang, G., Forsyth, D., and Hoiem, D. (2010). Comparative object similarity for improved recognition with few or nor examples. In *CVPR*.

[Wang and Zucker, 2000] Wang, J. and Zucker, J.-D. (2000). Solving the multiple-instance problem: A lazy learning approach. In *ICML*.

[Wang and Zhou, 2007] Wang, W. and Zhou, Z. H. (2007). Analyzing co-training style algorithms.

[Weisstein, 2002] Weisstein, E. W. (2002). *CRC Concise Encyclopedia of Mathematics*. Chapman & Hall, 2 edition.

[Weston et al., 2008] Weston, J., Ratle, F., and Collobert, R. (2008). Deep learning via semi-supervised embedding.

[Wolpert, 1992] Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

[Wolpert, 1996] Wolpert, D. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, pages 1341–1390.

[Xu et al., 2009] Xu, Z., Jin, R., Lyu, M. R., and King, I. (2009). Discriminative semi-supervised feature selection via manifold regularization.

[Yang et al., 2008] Yang, L., Jin, R., and Sukthankar, R. (2008). Semi-supervised learning with weakly-related unlabeled data: Towards better categorization. In *Advances in Neural Information Processing Systems*.

[Yilmaz et al., 2006] Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Comput. Surv.*

[Yu et al., 2008] Yu, K., Xu, W., and Gong, Y. (2008). Deep learning with kernel regularization for visual recognition. In *ECCV*.

[Zaki and Nosofsky, 2007] Zaki, S. and Nosofsky, R. (2007). A high-distortion enhancement effect in the prototype learning paradigm: Dramatic effects of category learning during test. *Memory & Cognition*, 35(8):2088 – 2096.

[Zeiler et al., 2010] Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *CVPR*.

[Zhang and Viola, 2008] Zhang, C. and Viola, P. (2008). Multiple-instance pruning for learning efficient cascade detectors. In *NIPS*.

[Zhang and Goldman, 2002] Zhang, M.-L. and Goldman, S. (2002). Em-dd: An improved multi-instance learning technique. In *NIPS*.

[Zhang and Goldman, 2001] Zhang, Q. and Goldman, S. (2001). Em-dd: An improved multiple instance learning technique. In *NIPS*.

[Zheng and Webb, 2005] Zheng, F. and Webb, G. I. (2005). A comparative study of semi-naive bayes methods in classification learning. In *The Fourth Australian Data Mining Conference (AusDM05)*, pages 141–156.

[Zhou and Li, 2005] Zhou, Z.-H. and Li, M. (2005). Tri-training. exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and DAta Engineering*, 17(11):1529–1541.

[Zhou et al., 2009] Zhou, Z.-H., Sun, Y.-Y., and Li, Y.-F. (2009). Multi-instance learning by treating instances as non-i.i.d. samples. In *ICML*.

[Zhou and Xu, 2007] Zhou, Z.-H. and Xu, J.-M. (2007). On the relation bewtween multi-instance learning and semi-supervised learning. In *ICML*, volume 43.

[Zhu and Ghahramani, 2002] Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical report.

[Zhu et al., 2003] Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*.

[Zhu and Goldberg, 2009] Zhu, X. and Goldberg, A. B. (2009). *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, third edition.

[Zhu et al., 2007] Zhu, X., Rogers, T., Qian, R., and Kalish, C. (2007). Humans perform semi-supervised learning too.

[Zou et al., 2008] Zou, H., Zhu, J., and Hastie, T. (2008). New multi-category boosting algorithms based on multi-category fisher-consistent losses. *Annals of Applied Statistics*.